Analysis of the Monte Carlo Calculations, 1947-8

Mark Priestley & Thomas Haigh
www.EniacInAction.com
January 2016, version 1.0

This report describes the evolution of the Monte Carlo calculations carried out on ENIAC during 1948. It provides technical background for the work we present in the paper "Los Alamos Bets on ENIAC" and in chapters 8 and 9 of the book *ENIAC In Action: Making and Remaking the Modern Computer* (MIT Press, 2016). [1]

We consider three main stages in the development of these calculations:

1. John von Neumann's original plan of computation, described in his letter of March 11, 1947 to Robert Richtmyer.
2. The Monte Carlo First Run program, run on ENIAC during April and May 1948.
3. The Monte Carlo Second Run program, run on ENIAC during September 1948.

Despite substantial elaboration, the basic physical model and the computational approach defined by von Neumann in 1947 were preserved in the calculations carried out in 1948. Starting with an examination of the original plan is therefore a good way to come to understand the complexities of the later programs, as well as the changes that were required to transform the plan into a program written in the modern code paradigm.[2]

Sections 1, 2 and 3 of this report describe the structure and development of the programs in these three stages. Section 4 provides an overview of how the programs relate to each other and Section 5 describes in detail the function and evolution of each main "region" of the computation from the original plan to the Second run program.

1 Von Neumann's 1947 computing plan

In a letter of March 11, 1947 to Los Alamos physicist Robert Richtmyer, John von Neumann discussed the use of "statistical methods to solve neutron diffusion and multiplication problems, in accordance with the principle suggested by Stan Ulam".[3] The aim was to model the paths taken by neutrons in a given material assembly, their collisions with other particles in that space, and the outcome of such collisions, including the production of further neutrons in fission events. A principal concern of this work was to model the progression of nuclear chain reactions.

1.1 The physical model

The simulation took place in a spherical space or, as von Neumann described it, a "spherically symmetric geometry". A consequence of, or perhaps a motivation for, this assumption was that a

---

[1] T. Haigh, M. Priestley, C. Rope, "Los Alamos Bets on ENIAC: Nuclear Monte Carlo Simulations, 1947-8", *IEEE Annals of the History of Computing*, 36:3 (July-September 2014):42–63. T. Haigh, M. Priestley, C. Rope,, *ENIAC In Action: Making and Remaking the Modern Computer* (MIT Press, 2016).

[2] This term is defined in our earlier paper: T. Haigh, M. Priestley, C. Rope, "Reconsidering the Stored-Program Concept", *IEEE Annals of the History of Computing*, 36:1 (January-March 2014):4–17.

[3] Von Neumann's letter and Richtmyer's reply made up the internal Los Alamos report "Statistical Methods in Neutron Diffusion" (LAMS-551 (Revised), April 9, 1947). The report was declassified in 1959 and reprinted in Von Neumann's Collected Works. The two letters were later reprinted on pages 149–155 of C. C. Hurd, "A Note on Some Early Monte Carlo Computations and Scientific Meetings", *Annals of the History of Computing*, 7:2 (April 1985):141–155.

neutron's position could be adequately modeled using only one spatial coordinate, namely its distance r from the center of the assembly.

The material composition of the space could vary. In his initial, informal description of the model, von Neumann specified:

> Variable (if desired, continuously variable) composition along the radius, of active material (25 or 49), tamper material (28 or Be or WC), and slower-down material (H in some form).[4]

For computational purposes, the possibility of varying material composition was modeled by a system of discrete *zones*:

> Describe the inhomogeneity of this system by assuming N concentric, homogeneous (spherical shell) zones, enumerated by an index $i = 1, \cdots, N$. Zone No. i is defined by $r_{i-1} \leq r \leq r_i$, the $r_0, r_1, r_2, \cdots, r_{N-1}, r_N$ being given
>
> $$0 = r_0 < r_1 < r_2 < \cdots < r_{N-1} < r_N = R,$$
>
> where R is the outer radius of the entire system.

"Homogeneous" here means "of uniform composition", not "a single type of material":

> Let the system consist of the three components … denoted A [active material], T [tamper material], S [slower-down material], respectively. Describe the composition of each zone in terms of its content of each of A, T, S. Specify these for each zone in relative volume fractions. Let these be in zone No. i x_i, y_i, z_i, respectively.

In order to calculate the probability of a collision, it is necessary to know the material cross-sections as well as the volume fractions. Cross-sections give the probability that an interaction of a certain type will take place in a given material, and are expressed as a function of velocity. Von Neumann's definition of these functions also made explicit the different outcomes that could result from a collision.

> Introduce the cross sections per cm^3 of pure material, multiplied by ${}^{10}\log e = .43\cdots$, and as functions of the neutron velocity v, as follows:
>
> Absorption in A, T, S: $\sum_{aA}(v)$, $\sum_{aT}(v)$, $\sum_{aS}(v)$.
>
> Scattering in A, T, S: $\sum_{sA}(v)$, $\sum_{sT}(v)$, $\sum_{sS}(v)$.
>
> Fission in A, with production of 2, 3, 4 neutrons: $\sum_{fA}^{(2)}(v)$, $\sum_{fA}^{(3)}(v)$, $\sum_{fA}^{(4)}(v)$.

A neutron can be absorbed by the nucleus it collides with, in which case it drops out of the simulation, or it can be scattered, in which case it will emerge from the collision following a possibly modified trajectory. Alternatively, a collision can lead to fission, in which case von Neumann assumed that 2, 3 or 4 new neutrons would result whose paths need to be followed

---

[4] Sic. 25, 28, and 49 were Los Alamos shorthand for uranium 235, uranium 238, and plutonium 239, respectively. WC is tungsten carbide, used by Los Alamos as a neutron reflector (tamper). The reference to "Be" is unclear. The *Los Alamos Primer*, a set of lecture notes used from 1943 in the induction of new recruits to Los Alamos, recommended the use of "the densest available materials (Au, W, Re, U)" as tamper (p. 7). However, beryllium (Be) is a very light element (atomic number 4) and so would appear to be a poor choice of tamper. It is possible that von Neumann intended to refer to rhenium (Re, atomic number 75).

further. When scattering or fission takes place, some assumptions about the behavior of the neutrons after the collision are specified:

> Scattering as well as fission are [sic] assumed to produce isotropically distributed neutrons, with the following velocity distributions:
>
> If the incident neutron has the velocity v, then the scattered neutrons [sic] velocity statistics are described for A, T, S, by the relations
>
> $$v' = v\varphi_A(v),\ v' = v\varphi_T(v),\ v' = v\varphi_S(v).$$
>
> Here v' is the velocity of the scattered neutron, $\varphi_A(v)$, $\varphi_T(v)$, $\varphi_S(v)$ are known functions, characteristic of the three substances A, T, S (they all vary from 1 to 0), and ν is a random variable, statistically equidistributed in the interval 0, 1.
>
> Every fission neutron has the velocity v_0.

1.2 Describing neutrons

The underlying idea of the application of Monte Carlo methods to neutron diffusion problems is to follow the progress of individual neutrons through the physical assembly, using pseudo-random numbers to determine the various events that affect a neutron during its lifetime. Von Neumann therefore went on to consider what information the model must maintain about individual neutrons in order to permit this:

> In this model the state of a neutron is characterized by its position r, its velocity v, and the angle θ, between its direction of motion and the radius. It is more convenient to replace θ by $s = r\ cos\ \theta$, so that $\sqrt{(r^2 - s^2)}$ is the "perihelion distance" of its (linearly extrapolated) path.
>
> Note that if a neutron is produced isotropically; i.e., if its direction "at birth" is equidistributed, then (because space is three-dimensional) $cos\ \theta$ will be equidistributed in the interval −1, 1; i.e., s in the interval $-r, r$.
>
> It is convenient to add to the characterization of a neutron explicitly the No. i of the zone in which it is found; i.e. with $r_{i-1} \leq r \leq r_i$. It is furthermore advisable to keep track of the time t to which the specifications refer.
>
> Consequently, a neutron is characterized by these data: i, r, s, v, t.

1.3 The numerical model

Von Neumann planned to hold the characteristics of an individual neutron on a punched card. As a result, whenever some of the data characterizing a neutron changed, and in particular when a collision took place, it was necessary to punch a new card or, as von Neumann phrased it, to "start a new neutron". An individual card therefore did not represent a neutron as such, but rather a certain period in the life of a neutron. Each card would hold the following data:

- C_1: i – the zone in which the neutron is found at time t
- C_2: r – the distance of the neutron from the center of the assembly at time t
- C_3: s – a parameter defining the direction of the neutron's path at time t
- C_4: v – the velocity of the neutron at time t
- C_5: t – the time to which the specifications refer

In addition, each card would also contain the following random values that would be required in the next step of the computation. Von Neumann did not specify how these numbers were to be generated, but each was to be a real number equidistributed in the range 0 to 1.

- R_1: λ – the probability of a neutron travelling a distance d^1 without a collision
- R_2: μ – used to determine what type of collision takes place
- R_3: ν – used in the calculation of a scattered neutron's new velocity
- R_4, R_5, R_6, R_7 : ρ', ρ'', ρ''', ρ'''' – used to determine the position of neutrons after scattering and fission

Finally, the calculation made use of the following "global" data and functions representing aspects of the physical environment that the neutrons were travelling in:

- r_i, the zone radii, and x_i, y_i, z_i, the density of the three types of material (active, tamper and slower-down material, respectively) in zone i.
- $\sum_{aA}(v)$, $\sum_{aT}(v)$, $\sum_{aS}(v)$, $\sum_{sA}(v)$, $\sum_{sT}(v)$, $\sum_{sS}(v)$, $\sum_{fA}^{(2)}(v)$, $\sum_{fA}^{(3)}(v)$, $\sum_{fA}^{(4)}(v)$ – the cross-sections. Nine functions of v, for the relevant combinations of event type and material;
- v_0 – the velocity of new neutrons created by fission;
- $\varphi_A(v)$, $\varphi_S(v)$, $\varphi_T(v)$ – functions determining neutron velocity after scattering;
- $-{}^{10}\log \lambda$ – a function of the random variable λ.

Von Neumann specified that the values of the functions r_i, x_i, y_i, z_i should be tabulated for the different values of i. The other functions had continuous domains, however, and he left it undecided whether their values should be tabulated, or calculated by interpolation or by polynomial approximation.[5]

1.4 The computing plan

After defining the data characterizing a neutron, von Neumann described how the neutron described by the data held on an individual punched card could be tracked:

> Now consider the subsequent history of such a neutron. Unless it suffers a collision in zone No. i, it will leave this zone along its straight path, and pass into zones Nos. $i+1$ or $i-1$. It is desirable to start a "new" neutron whenever the neutron under consideration has suffered a collision (absorption, scattering or fissioning – in the last-mentioned case several "new" neutrons will, of course, have to be started), or whenever it passes into another zone (without having collided).

By "new" neutron, von Neumann meant the production of a card with an updated set of neutron characteristics on it. Processing the data on an input card would lead to between one to four new cards being produced. Cards with $v = 0$ (indicating that the neutron had been absorbed) or $i = N + 1$ (indicating that the neutron had "escaped", or travelled outside the physical range of the simulation) would be discarded, and all other cards would be run through the calculation again until sufficient data about the progress of the reaction being modeled had been obtained.

The calculations were described in a "tentative computing sheet" given as a sequence of 81 individual steps. Each step was numbered, and consisted of an "instruction" to be carried out at that step, and an "explanation", which described the value that had been computed in terms of the variables used in the mathematical description of the problem.

The majority of the steps (68 out of 81) specified an arithmetic operation to be carried out. The operands could be values from the input card or the results of previous steps. The number of the step at which a value was obtained was used to refer to it in later steps. The simple operations used

---

[5] In a letter of March 27, 1947 to Stan Ulam (held in the Stanislaw M. Ulam Papers, American Philosophical Society, box series I-29) von Neumann reported on an ENIAC set-up for the plan, produced by Herman and Adele Goldstine, which "(assuming third-order polynomial approximations for all empirical functions), will exhaust about 80-90 percent of the ENIAC's programming capacity".

were addition, subtraction, multiplication and division, doubling a number, and the formation of its square and square root. 18 of these steps specified the evaluation of a function value, but von Neumann left it open at this stage how this would be done. At most one arithmetic operation was performed in each step, with the partial exception of step 14 which computed a square root and possibly changed its sign.

The steps were performed in numerical order, with conditional execution being provided for by a mechanism of "predicates". Four steps (numbers 6, 9, 47 and 53) defined predicates based on previously calculated data values. Three steps (numbers 10, 11 and 48) represented a form of conditional expression which used a predicate to choose between alternative values. In steps 10 and 11, this mechanism was simply used to select one of two numbers, but in step 48 the selected value was then used as an operand in a division operation. In addition to this, 31 of the steps were prefixed with informally specified Boolean combinations of predicates indicating the circumstances under which those steps should be carried out.

Finally, six steps (51, 60, 69, 73, 77 and 81) specified the values of the neutron characteristics that would be printed on a card for various outcomes of the calculation.

To discriminate between these various outcomes, the computing plan proceeded through the following stages.[6]

1.4.1 C Calculate distance to zone boundary

Steps 1 – 15

The plan first establishes what will happen if the neutron continues along its existing trajectory and no collision takes place or, as von Neumann put it:

> Consider first, whether the neutron's linearly extrapolated path goes forward from zone No. i into zone No. $i+1$ or $i-1$.

Steps 1 – 11 of the computing sheet determine which of these possibilities occurs. Predicate A represents the case of the neutron moving outwards, in which case the next zone must be $i+1$; predicate B represents the case of the neutron moving inwards. If it is moving inwards, it could still end up in zone $i+1$, if its path does not take it across the inner boundary of zone i; this case is denoted by predicate B', and the case where the resulting zone is $i-1$ by predicate B''.

Step 10 defines r^*, the neutron's new position, to be either the inner or outer boundary of its current zone, depending on its direction of movement. Step 11 defines the value of a variable ε to be +1 if the neutron is moving outwards, and –1 if it is moving inwards; in step 50, this value will be used to update the index of the neutron's current zone.

Steps 12 – 15 then calculate s^*, the neutron's direction of travel at point r^*, and d, the distance from the neutron's current position to its exit point from its current zone.

1.4.2 E Determine if terminal event is collision or escape

Steps 16 – 47

[6] The letters used to identify the stages in the following description were first used in an overview flow diagram of the First run program: see undated manuscript page numbered "0," in John von Neumann papers, Library of Congress, Washington, D.C., box 11, folder 8. (This collection is subsequently referred to as JvN-LoC.) They are used here to enable cross-references to be made between the different stages in the computing plan and First and Second run programs. A table summarizing the development of the program is given in Section 4.

The next stage in the calculation determines whether the neutron will in fact travel the distance to the zone boundary, or whether it will suffer a collision before that point is reached. Von Neumann commented that "[i]t is at this point that the statistical character of the method comes into evidence".

Von Neumann defines "the probability that the neutron will travel a distance d^1 without suffering a collision" as $\lambda = 10^{-f d^1}$, where d^1 represents the distance from the neutron's current position to the point at which it *does* suffer a collision, and f is the sum of the cross-sections multiplied by the material density in the current zone. f is calculated in steps 16 – 44 of the plan, and in the course of forming it, partial sums $f_1 \cdots f_6$ are stored for use in step 53 to determine the type of collision that has occurred.

The randomly-assigned value of λ is given on the input card, and steps 45 and 46 then calculate d^1 as $(-\,^{10}\log \lambda)/f$. Step 47 then compares d^1 with d, the distance to the current zone boundary. If $d^1 \geq d$, the distance to the collision is greater than the distance to the zone boundary, and so the neutron escapes from its current zone before suffering a collision; step 47 denotes this outcome, known later as "zonal escape", by the predicate P. If on the other hand $d^1 < d$, the collision takes place before the neutron leaves its current zone; this outcome is denoted by the predicate Q.

1.4.3 N Zonal Escape

Steps 48 – 51

If the neutron reaches its zone boundary without a collision, predicate P holds, and steps 48 and 49 use the distance d to calculate t^*, the time of the terminal event.[7] Step 50 uses the value of ε from step 11 to calculate the new zone index i^*, and in step 51 the updated characteristics of the neutron are summarized for printing. The computation for the current neutron then ends.

1.4.4 H Determine collision type

Steps 48, 49, 52 – 59

If a collision has taken place before the neutron reaches the zone boundary, predicate Q holds, and steps 48 and 49 use the distance d^1 to calculate t^*. Steps 52 and 53 then determine the nature of the collision. There are seven possibilities: the neutron can be absorbed, it can be scattered in one of the materials A, T or S, or a fission event can take place in material A leading to the production of 2, 3 or 4 new neutrons. The probabilities of these outcomes are given by products of their cross-sections with the amount of material present, and are represented by the intervals $[0, f_1)$, $[f_1, f_2)$, $\cdots$, $[f_5, f_6)$, $[f_6, f]$. To determine the nature of the collision, the random variable μ is multiplied by f; the product μf then lies in one of the subintervals of $[0, f]$. The outcome is denoted by a predicate Q_i, where $1 \leq i \leq 7$.

Steps 54 – 59 then compute r^*, the position of the collision. This will be the same for all types of collision.

1.4.5 L Absorption

Step 60

In the case of absorption (predicate Q_1), the neutron has disappeared. This situation is modeled by setting the new velocity equal to 0. Step 60 lists the contents of the output card for this case.

1.4.6 J & K Scattering

[7] These steps use an intermediate variable τ. This may have been to conform to an (unstated) rule that only one arithmetic operation should be performed in each step of the calculation

Steps 61 – 69

In the case of scattering (predicates Q_2, Q_3 and Q_4), the neutron's new velocity and direction are calculated in steps 61 – 69. The new velocity is calculated as $v' = v\varphi_A(v)$ (or φ_T, φ_S, depending on material type), where v is a random variable read from the current card, and the new direction is calculated in steps 66 – 68 using the random variable ρ'. The new neutron characteristics are printed in step 69.

1.4.7 L Fission

Steps 65 – 81

In the case of fission producing 2, 3 or 4 new neutrons (predicates Q_5, Q_6 and Q_7), all the successor neutrons have the same velocity, v_0, set in step 65, but different directions. The first new neutron is considered equivalent to a scattered neutron, and its new direction calculated in steps 66 – 68. Further random variables ρ'', ρ''', and ρ'''' are then used to compute the directions for the remaining fission products, the computations being repeated and new cards printed in steps 70 – 73, 74 – 77 and 78 – 81.

1.5 Operating procedure

Von Neumann wrote of his plan that

> It is, of course, neither an actual "computing sheet" for a (human) computer group, nor a set-up for the ENIAC, but I think that it is well suited to serve as a basis for either.

He had, however, clearly given the possibility of using ENIAC serious consideration, writing earlier in the letter that "the problem ..., in its digital form, is well suited for the ENIAC".

In either case, the mathematical procedure described by the plan would be part of a larger computational process that would still include significant amounts of manual processing. Von Neumann made the following assumption about the scale of the computation:

> Assume that one criticality problem requires following 100 primary neutrons through 100 collisions (of the primary neutron or its descendants) per primary neutron.

The initial stack of 100 cards would each be punched with the five numbers characterizing a neutron together with the seven random numbers required in the course of modeling its progress. As this stack was fed through ENIAC, a new stack of cards would be punched, each containing the data for a "descendent" neutron. Some of these would not need to be followed further:

> the cards with $v = 0$ (i.e. corresponding to neutrons that were absorbed within the assembly) as well as those with $i = N + 1$ (i.e. corresponding to neutrons that escaped from the assembly), may be sorted out.

The cards that were reserved for further processing contained only the five neutron characteristics, but von Neumann notes that:

> The 7 random variables can be inserted in a subsequent operation.

At the end of the 100 iterations of these processes, the resulting (potentially very large) stack of cards output would be subjected to statistical analysis to determine properties of the reaction being modeled:

> The manner in which this material can then be used for all kinds of neutron statistic investigations is obvious.

In other words, von Neumann's plan describes only the central mathematical operations involved in following a single neutron's trajectory. These would sit at the heart of a much larger computational system which would make use of other devices, such as IBM punched card equipment, as well as relying significantly on human labor to move the stacks of cards around.

The plan makes a very clear separation between the computational algorithm and the numerical data characterizing a particular physical configuration. This was a deliberate choice:

> A common set-up of the ENIAC will do for all criticality problems. In changing over from one problem of this category to another one, only a few numerical constants will have to be set anew on one of the "function table" organs of the ENIAC.

The phrase "common set-up" here suggests that von Neumann was envisaging the use of ENIAC in its original, locally programmed, mode. In this connection, it is also worth noting that the computing sheet does not specify the flow of control in the style of a flow diagram or an EDVAC-style order code. Rather than describing various computational paths, it presents a linear sequence of operations whose individual execution is governed by the truth or falsity of a number of predicates.

A final comment gave some very provisional space and time estimates for the computation on ENIAC.

> I cannot assert this with certainty yet, but it seems to me very likely that the instructions given on this "computing sheet" do not exceed the "logical" capacity of the ENIAC. I doubt that the processing of 100 "neutrons" will take much longer than the reading, punching and (once) sorting time of 100 cards, i.e., about 3 minutes. Hence, taking 100 "neutrons" through 100 of these stages should take about 300 minutes, i.e., 5 hours.

A couple of weeks later, as he reported to Stan Ulam, he felt in a position to make this estimate more precise:[8]

> The computational set-up which I sent you has been investigated more carefully from the ENIAC point of view by H. H. and A. K. (Mrs.) Goldstine. They will probably have a reasonably complete ENIAC set-up in a few days. It seems that the set-up, as I sent it to you (assuming third-order polynomial approximations for all empirical functions), will exhaust about 80-90 percent of the ENIAC's programming capacity.

2 The First run calculations

The set-up produced by the Goldstines was never put onto ENIAC, however. During 1947, a system of "central control" was developed for ENIAC, enabling it to execute programs written in an EDVAC-style order code, and subsequent development of von Neumann's plan took place within this framework. [9] This development culminated in what became known as the "First run", the execution of the program on ENIAC during April and May, 1948.

The John von Neumann papers in the Library of Congress contain a significant amount of material documenting the development of the first Monte Carlo program, which ran on ENIAC in April and May 1948. The main stages of the development are the following:

---

[8] J. von Neumann, letter to S. M. Ulam, March 27, 1947. Stanislaw M. Ulam papers, American Philosophical Society, Ms. Collection 54, series I.

[9] For more details on the central control, see T. Haigh, M. Priestley, C. Rope, "Engineering the `Miracle of ENIAC': Implementing the Modern Code Paradigm", *IEEE Annals of the History of Computing*, 36:2 (April-June 2014):41–59.

- At the beginning of April 1947, Richtmyer responded to von Neumann's proposals with a number of detailed comments and suggestions.[10]
- Once the decision was taken to develop the program for the centralized control system, von Neumann produced a complete flow diagram incorporating most of Richtmyer's proposals.[11] This diagram adheres closely to the notation presented in the report on programming techniques that von Neumann and Herman Goldstine had recently issued.[12]
- Subsequent development included a number of significant decisions about the representation of data and the algorithms to be used. A substantial amount of material has been preserved, allowing us to reconstruct many of the details of this work.
- A new flow diagram, incorporating all these modifications, was produced in December, 1947.[13] The accuracy of this diagram as documentation of the actual First run program is attested to by the high degree of consistency between it and the later description given in the typescript *Actual Running of the Monte Carlo Problems on the ENIAC*, which describes both the First and Second run programs.[14]

As well as many differences of detail between the First run program and von Neumann's original plan, the development introduced three more significant modifications to the overall organization of the computation.

2.1 Census

The most substantial change was motivated by the need to ensure that the cards produced by the simulation were suitable for the planned statistical analyses. Von Neumann had originally proposed following the fortunes of an initial deck of 100 neutrons through 100 computational cycles. Richtmyer observed that at the end of this procedure, the "multiplicative chains" of events would have taken different lengths of time, depending on the life histories of individual neutrons. Let t_1 be the earliest and t_2 the latest time at which a chain of simulated events had terminated. He then pointed out that:

> if one wishes, for example, to find the spatial distribution of fissions, it would be natural to examine all fissions occurring in some interval Δt and find their spatial distribution.

However, as a result of this procedure, we only know about *all* the fission events for the period preceding t_1, and there is no guarantee that the total spatial distribution of fissions after t_1 will be the same as those that the simulation happens to have considered. In this case, all the work done

---

[10] Richtmyer's comments were contained in a letter to von Neumann of April 2, 1947. The letter is reprinted on pages 154–155 of C. C. Hurd, "A Note on Some Early Monte Carlo Computations and Scientific Meetings", *Annals of the History of Computing*, 7:2 (April 1985):141–155. All quotations attributed to Richtmyer in this section are taken from this letter.

[11] Ten undated manuscript pages in John von Neumann's handwriting, numbered I, II.a – II.g, III, IV: JvN-LoC box 11, folder 8. The flow diagram appears on pages II.a – II.g, and is referred to in the rest of this report as "von Neumann's draft flow diagram".

[12] H. H. Goldstine and J. von Neumann, Planning and Coding of Problems for an Electronic Computing Instrument, Part II, Volume 1, Institute of Advanced Study, 1 April, 1947.

[13] Flow diagram headed "MONTE CARLO" and dated "12/7/1947": JvN-LoC, box 11, folder 8.

[14] *Actual Running of the Monte Carlo Problems on the ENIAC*, undated manuscript in Klara von Neumann's handwriting and typescript with handwritten corrections, JvN-LoC, box 12, folder 6. A letter from Nick Metropolis to Klara von Neumann, dated September 23, 1949, appears to refer to these documents (JvN-LoC, box 19, folder 7). Metropolis writes: "Here is your manuscript together with a rough typewritten copy. You will observe that the first several pages have some corrections on them which I started to make". In the remainder of this report this document is referred to as *Actual Running*; quotations are taken from the corrected typescript and cited as "(AR:3)", for example.

following events after t_1, potentially a large fraction of the total computation, would have been wasted.

Richtmyer's solution to this problem was the following:

> The obvious remedy for this difficulty would seem to be to follow the chains for a definite time rather than for a definite number of cycles of operation. After each cycle, all cards having t greater than some preassigned value would be discarded, and the next cycle of calculation performed with those remaining. This would be repeated until the number of cards in the deck diminishes to zero.

The First run implemented this suggestion by taking a periodic "census" of the neutrons in the simulation. The passage of time was divided into a number of "census cycles": before any cards were read, a "census time" was set, and the chains of events were only followed up to that point. The stack of cards was repeatedly processed until the only surviving neutrons were the so-called "census cards" corresponding to neutrons which had neither been absorbed nor escaped from the assembly before the census time. Then,

> [t]o keep an exact one hundred cards for each census interval, an appropriate number of the census cards were either duplicated by a random choice, if the stack had decreased, which happened normally (but of course not necessarily or exclusively) when the assembly was under-critical, or discarded if the stack had increased, which was similarly related to the assembly being critical or super-critical. These cycles, i.e. census time intervals, could be continued as many times as desired, using each time the census cards, obtained from the end of the previous cycle, as input data. (AR:5)

By following the progress of all neutrons up to a census time, this procedure ensured that all fissions in that time period had been considered, thus addressing Richtmyer's original concern Readjusting the deck of cards at census times ensured that a reasonably consistent neutron population, sufficiently large to generate meaningful statistics, was preserved throughout the simulation.

Implementing the census scheme made the First run program much more complex than von Neumann's original plan, which had only described the calculations that would be required to track the course of an individual neutron through one stage in its trajectory. Rather than being a simple sequence of instructions, the program's basic structure now consisted of three nested loops, described in more detail below. The current census time was set on the ENIAC's constant transmitter and once all neutrons in the current stack had reached the census time, it was manually updated and a new census cycle started.

2.2 Velocity intervals

A second substantial modification in the First run program involved the handling of the neutrons' velocity. New neutrons, both those new to the simulation and also the products of fission events, were specified to have

> a random velocity distributed over the fission spectrum, represented by the "centers of gravity" $V(\xi)$, $(\xi = 0, 1, \cdots, 9)$ of an appropriate subdivision of that spectrum into suitable velocity intervals. (AR: 4)

A page of notes associated with an early version of the First run flow diagram contains the following table:[15]

[15] Ten undated manuscript pages in John von Neumann's handwriting, numbered I, II.a – II.g, III, IV: JvN-LoC box 11, folder 8. The table reproduced in the text is on p. I. Pages II.a – II.g give a complete flow diagram for the Monte Carlo computation, and on pages III and IV von Neumann works out an estimate of the running time of the program.

$k =$	0	1	⋯	8	9
	$V \le V_1$	$V_1 \le V \le V_2$	⋯	$V_8 \le V \le V_9$	$V_9 \le V$
	V_0^*	V_1^*	⋯	V_8^*	V_9^*

This shows that ten velocity intervals were defined by nine boundary values, V_i, the first and last intervals being unbounded below and above, respectively. The final row shows the "centers of gravity", later designated as values of a function v. When it was necessary to allocate a random velocity to a new neutron, instead of trying to select a random value from the whole velocity spectrum, a random digit was used to select a velocity interval, and the center of gravity of that interval used as the neutron's velocity.

If a neutron was inelastically scattered, it acquired a new velocity which could be any value in the spectrum, not just one of the centers of gravity. For these neutrons, and also for the new neutrons, a new step was introduced early in the computation to work out which velocity interval the neutron fell into.[16]

We have not found an explicit account of the motivation for the introduction of velocity intervals. Our hypothesis is that it is connected with the approach taken to store the cross-sections for the different types of material in the simulation. Von Neumann had originally specified that the cross-sections were given by functions of a neutron's velocity, such as $\sum_{aA}(v)$, and a footnote stated that these functions were to be

> Tabulated, to be interpolated, or approximated by polynomials. (Continuous domain.)

In the First run, these functions were tabulated on one of the ENIAC's function tables, as described below, one value being stored for each of the ten velocity intervals. No interpolation was carried out, so it would be sufficient to know, for each neutron, which interval its current velocity fell into. We do not know if the possibility of interpolating more accurate values had been considered. If it had, it may have been ruled out because of restrictions on code or function table size, or computation time, or because it was felt that sufficient accuracy could be obtained by simply using the "center of gravity" values.

2.3 Pseudo-random numbers

In the First and Second runs, the random numbers that were required at various points in the simulation were generated on demand instead of being pre-computed and punched onto the neutron cards. Von Neumann explained the motivation for this as follows:

> Our main reason for using these arithmetical pseudo-random numbers was that we intend to solve the Monte Carlo problems on an electronic automatic computing machine: the ENIAC. If hand calculation, or calculation on some slower automatic device is contemplated, then the arithmetical-pseudo-random procedure does not seem reasonable to me. It is possible to obtain large quantities of "guaranteed" random numbers recorded in tables or on punch-card … The only point is that the ENIAC is much faster in producing a number by some arithmetical function $f(x)$ … than any reading it from a punch-card. In other words, it is preferable to explain to the ENIAC how to produce its random numbers internally and mathematically than to furnish it with a ready-made list of random numbers which it has to read.[17]

---

[16] For a new neutron, the velocity interval could have been directly derived from the random digit used to assign it a velocity, but we have found no record of this approach being adopted in either the First or Second run.

[17] J. von Neumann, letter to A. S. Householder, February 3, 1948. Reprinted in M. Rédei (ed.), *John von Neumann: Selected Letters*, American Mathematical Society (2005), 141–3.

The First run flow diagram contains a separate operation box which was invoked at different points in the computation to generate a new pseudo-random number when required:

> Although lists of the checked random numbers had been available at the start of either run, it was found more convenient and faster to have the squaring, i.e. "refresh" done by the ENIAC. The "refresh" random number is shown on a separate block on the flow-diagrams, with ρ indicating the places where a new random number was needed and ω_n the various exits to the particular path where the computation had to be continued after obtaining the new random number. (AR:22)

The separate box describes the algorithm used to refresh the random number:

> A randomly chosen n digit number ξ was taken as the base ($n = 8$ in the First run, $n = 10$ in the Second run), squared and the middle n digits used as, on one hand, the next supply of random digits, as needed in the course of the computation and, on the other hand, as the base of the next number to be squared. (AR:21)[18]

Although reference is occasionally made to a "random number equidistributed between 0 and 1" (AR:17), in practice ξ was usually treated as a string of random digits from which substrings could be extracted to provide, for example, a random digit between 0 and 9.

After the event, Von Neumann commented that:

> Both groups of numbers, the 10 digit one and the 8 digit one, were used in "Monte Carlo" type problems on the ENIAC, and they seem to be all right.[19]

The statistical randomness of the numbers generated by this technique had been tested by hand, but only up to a certain point:

> Since, however, in the first case [run], only two thousand and in the second case [run], little over three thousand squarings were statistically tested for randomness, provisions were made to recommence with the original random numbers each time when, in the course of the calculations two (three) thousand squarings have already been made. Having reached this point, the machine was instructed to read a "special" card, containing the original random number and to place it in the accumulator assigned as, more or less, permanent memory location for the changing value of ξ. (AR:21–22)[20]

2.4 The physical model

The physical model assumed for the First run calculations was essentially the same as that of the original plan. The spherical space was divided into a number of concentric zones. The total number of zones was denoted by I, and their boundaries by r_i, $0 \leq i \leq I$. Eleven positions were allocated in ENIAC's numeric function table to store the boundary values, so the maximum number of zones that could be modeled in the First run was ten.[21]

---

[18] The archival materials we have examined do not make clear how the restriction to 8 digits was enforced in the First run. All the versions of the refresh algorithm preserved from both First and Second runs are essentially the same, and extract 10 digits from the product formed by squaring ξ. We have found no evidence of additional coding to reduce this to 8 digits. Furthermore, an undated page of notes in JvN-LoC, box 11, folder 8, contains a list of "Parts of ξ used" referring to all 10 digit positions. This document is mostly consistent with the December 1947 flow diagram, which at various points specifies the extraction of individual digits from 9 of the 10 positions in the accumulator holding ξ (position 5 is not referred to in the flow diagram).

[19] J. von Neumann, letter to C. C. Hurd, 3 December, 1948. Reprinted in M. Rédei (ed.), *John von Neumann: Selected Letters*, 144–5.

[20] A different technique was used in the Second run to restart the sequence of pseudo-random numbers.

[21] For the storage allocation, see an undated manuscript sheet in JvN-LoC, box 11, folder 8, headed "Function Table III Numeric".

Von Neumann had originally proposed three categories of active (A), tamper (T) and slower-down (S) material, and had specified that fission could only take place in active material. Richtmyer responded that

> in systems of interest to us, there will be an appreciable number of fissions in the tuballoy of the tamper, as well as in the core material.

He also commented that for some systems material S could be omitted.

In the First run, the modeling of material types was simplified in the light of these comments. Instead of the categorization into A, T and S, the model allowed for up to four different types of material, indexed by the variable $b = 0,1,2,3$, and made no assumption about what events could take place in each material, thus giving a greater freedom to model particular physical systems. The composition of each zone was described by giving the density of each type of material in each zone, the density of material b in zone i being denoted by f_i^b.

Richtmyer also noted that

> For metal systems of the type considered, it would probably be adequate to assume just one elastically scattering component and just one inelastically scattering component. These could be mixed with the fissionable components in suitable proportions to mock up most materials of interest.

This introduced two new types of collision, elastic and inelastic scattering, to join the absorption and fission events that von Neumann had initially considered. In the First run, these different collision types were identified by the values of a variable c (0 = elastic scattering, 1 = inelastic scattering, 2 = fission, 3 = absorption).

The collision cross-sections depended on the three parameters of material type, collision type, and neutron velocity. Von Neumann had enumerated seven relevant combinations of material and collision, and so defined seven distinct cross-sections functions, such as $\sum_{aA}(v)$. However, Richtmyer pointed out that

> we are not likely for some time to have data enabling one to distinguish between the velocity dependence of the three functions
>
> $$\sum_{fA}^{(2)}(v), \sum_{fA}^{(3)}(v), \sum_{fA}^{(4)}(v)$$
>
> that you introduce so that for any particular isotope these might as well be combined into a single function of velocity with a random procedure used merely for determining the number of neutrons emerging.

Exactly this procedure was followed in the First run, where the cross-section functions were tabulated for each of the 160 possible combinations of material type b, process c, and neutron velocity interval k.

As von Neumann had suggested in the letter to Richtmyer, the data characterizing the physical set-up of the simulation was stored in a single function table and could be altered according to the demands of specific problems:

> For all six problems, in the First Monte Carlo run, one common Flow-Diagram and one common code, i.e., logical sequence was used. Accordingly, only the one Function Table containing the numerical data specific to the assembly to be computed, had to be changed from one problem to another. (AR:7)

ENIAC's function tables had 104 rows each containing 12 digits and two signs. This enabled two signed six digit numbers, usually referred to as the A (left) and B (right) numbers, to be stored in

each row. Work on planning the allocation of data and estimating the amount of storage required began early in the development process. The same document that described the introduction of the velocity intervals, referred to in Section 2.2 above, also contained the following initial allocation of numeric material:[22]

Fc. T. Numeric Capacity used:

$S_c(h)$: 40	$V(\xi)$: 10 (R)	p_1^b	12 (R)
V_k : 5	$c(\xi)$: 10 (R)	p_2^b	
f_i^b : $4I$ (L)	r_i : I (R)	p_3^b	

Total so far: $45 + 4I$ (plus $32 - 3I$ (R)).

A later document gives a more complete table, with details of the allocation of data to function table rows.[23] The data and its allocation can be summarized as follows, though the document shows evidence of ongoing revision to the details of the allocation and the row numbers given here are not necessarily those used in the actual First run program.

- $S_c(h)$, the values of the cross-section functions, were stored in rows 0 – 39. The values of b (material type) and k (velocity interval index) were combined into a value h, where $0 \leq h \leq 39$. Then for each of the four values of c (process type), the values of $S_c(h)$ were stored in row h of the function table. Cross-section values were therefore held to three significant figures.
- V_k, the velocity interval boundaries for $0 \leq k \leq 9$, were stored in rows 40 – 45. Two values were held on each row, as $-V_k$, and the value of V_5 was repeated. The details of this storage allocation were chosen to simplify the algorithm used to locate the correct velocity interval for a neutron; the details of both the algorithm and the storage layout changed in the course of the development of the First run program.
- The 40 values of f_i^b, the density of material b in zone i, where $0 \leq b \leq 3$, were stored in the A field of rows 46 – 85.
- Four values $10^{-1} + \mu_b$ were stored in the A field of rows 86 – 89. These related to the mass of material b, and were used to distinguish the special case of elastic scattering in light materials.
- Ten values $V(\xi)$, the "centers of gravity" of the 10 velocity intervals, were stored in the B field of rows 46 – 55. These were used to allocate a random velocity, selected using a random digit ξ, to a neutron generated as the product of a fission event.
- Ten values $c(\xi)$ were stored in the B field of rows 56 – 65. These were precomputed values of $\cos(\pi/10(\xi + 0.5))$, $\xi = 0, 1, \cdots, 9$, used in the calculation of a neutron's velocity after elastic scattering had occurred.
- The zone boundaries r_i were stored in the B field of rows 66 – 76. Space was allocated for the storage of up to eleven values $r_0 < r_1 < \cdots < r_9 < r_{10}$.
- Sixteen values $-p_q^b$ were stored in the B field of rows 77 – 92. p_q^b represented the probability of fission in material b giving rise to q neutrons, $0 \leq q \leq 3$.
- Four values $-E_b$ were stored in the B field of rows 93 – 96. This parameter was used in the calculation of the neutron's new velocity in the case of inelastic scattering.
- I, the total number of zones, was stored in the B field of row 97.

[22] Undated manuscript page numbered "I.", in John von Neumann's handwriting: JvN-LoC box 11, folder 8.
[23] Undated manuscript sheet headed "Function Table III Numeric": JvN-LoC, box 11, folder 8.

- A few constants used in the course of the program were also stored in the remaining space in the numeric function table.

2.5 Describing neutrons

In line with the increasing complexity of the computational process, considerably more neutron-specific data was held on the cards in the First Run; apart from anything else, the decision to calculate random numbers on demand rather than include them on each card meant that there was more space available.

The physical characteristics of individual neutrons were represented by the same five parameters as in von Neumann's plan, but it was made clearer that the data related to a specific point in time. A parameter α was introduced,

> indicating the event that terminates the period of life (of the neutron) to which the card refers. (AR: 1)

The possible values of α in the First Run were:[24]

- $\alpha = 3$: Absorption;
- $\alpha = 4$: Total escape;
- $\alpha = 5$: Fission;
- $\alpha = 6$: Census;
- $\alpha = 7$: Special cards used to reset the sequence of random numbers (see above).

As before, i, t, r and v represented respectively the zone and time where the event took place, and the neutron's position and velocity at that time. However, the direction parameter $s = r\cos\theta$ was replaced by a new parameter $a = r(1 - \cos\theta)$.

In addition to these physical parameters, each card included four values which were used to help keep track of the life histories of neutrons. Each card was given a serial number $\bar{n}$, and the card also held the serial number of the "parent" and "ancestor" cards, denoted by n' and n^* respectively; these represented "the card from which the current card originated" (AR:3) and its ultimate originating card. Additionally, the parameter g represented the "fission generation" of the card, namely the number of fission events that involved in its history. These values were not used in the calculations, but were recorded on each card to help with subsequent indexing and analysis.

If the terminal event was fission, a separate card was not printed for each resulting neutron. Instead, each card contained a value q

> indicating the number of neutrons that originated in the terminal event and that are initiated by the card now under consideration; it is essentially its weight, i.e. the instruction to the machine how many times to use the data from this card and follow its course to the terminal event, before reading the data from the next card in the stack. $q = 1$ for each event, except if the collision was fission, when it may be any integer $\geq 1, \leq 4$, according to the number of neutrons created by fission – this number being determined by a random process, as provided by the Monte Carlo method. (AR:3)

The organization of the computation ensured that the data from such cards would be used repeatedly, giving rise to the appropriate number of successor cards This illustrates how increasing

[24] It's not clear why the values 0, 1 and 2 weren't used. An undated manuscript page from JvN-LoC box 11, folder 8 documents the mapping from the "abstract" storage locations used in von Neumann's initial flow diagram to a definite allocation of data values to ENIAC's accumulators, and among other things shows, with no explanation, α values of 0, 1, 2, and 3 of being mapped onto the new values 3, 4, 5 and 6.

the logical complexity of the First Run program enabled a simplification in the physical representation of the problem, by reducing the number of cards used.

The actual storage of this data on punched cards is described in box 47* of the December 1947 flow diagram, and is summarized in the table below. In describing the calculation, the flow diagram has to refer to both the original and the newly calculated values of many variables, and uses asterisks to indicate the newly calculated values.

Field	Print	Columns	Read
1	i	9–10	i
2	t^*	11–15	t
3	v	21–25	v
4	a^*	31–35	a
5	r^*	36–40	r
6	ξ^*	41–50	Not read
7	g^*	54–56	g
8	$\bar{n}$	61–65	n'
9	n^*	66–68	n^*
10	n'	69–74	Not read
11	α^*	75	α
12	q^*	76	q
13	q	80	Not read

In general, the table shows that newly calculated values (with asterisks) are printed at the end of the processing of a card, and that these are then read in as initial values (without asterisks) when the card is next read. Fields 1 – 5 hold the physical characteristics of the neutron.[25] Field 6 holds the most recently calculated value of the random number ξ which was printed on each card, but not subsequently read; presumably this would enable subsequent manual checking of the results recorded on the card, if required. Fields 7 – 9 hold the identifying data of the card: the serial number $\bar{n}$ of each card is printed in field 8, and the next time the card is processed is read in as the serial number of the parent card, and subsequently printed for reference in field 10. By contrast, the number of the ancestor card n^* is copied from card to card as the simulation progresses. Finally, field 13 shows that the card's previous weight is stored, but not used in the computation.

2.6 Operating procedure

The basic operating procedure of the First Run was to take an initial deck of 100 cards, as originally proposed by von Neumann, and repeatedly process them until all the surviving neutrons had reached the end of the current census period.

> To start any of the problems of the First Run, one hundred cards were read, each representing one neutron originated by fission (AR:4)

The initial neutrons were not necessarily the result of fission reactions; rather, the cards recorded the previous terminal event as fission so that the neutrons would automatically be given a randomly allocated velocity.

> All the initial neutrons were started at the center of the spherical assembly at zero time. These neutrons then, in turn, produced other neutron cards, indicating the event that terminated the calculation of their path, one of the

[25] It's not clear why the "Print" column showed i and v in place of the expected i^* and v^*. Reserving two digits for i suggests that the current zone may have been stored as a number in the range 1 – 10 rather than 0 – 9.

> events being that a census time T, has been reached. Total escape and absorption cards were then sorted out and removed from the newly printed stack since their path did not have to be followed any longer. Fission cards, representing neutrons which produced two or three new neutrons, (n.b. In the problems considered here only two or three way fission was considered) and whose time t was still within census time, were then put back in the reader until each newly printed card was either a total escape, absorption or census card, indicating that all neutrons that survived have reached the end of the census time interval. At this point, since there were no more cards to be read, the automatic iterative process of the machine was interrupted, the machine stopped and a new cycle was started with a new value for T, the next census time. (AR:4-5)

This suggests that total escape and absorption cards would never be read by the First Run program. However, the December 1947 flow diagram shows that these cards were read and then discarded, as described by a later passage in *Actual Running*:

> A card, representing a neutron, was first examined for α, i.e. the type of terminal event which occurred at its previous period of life. If the event was Absorption, ($\alpha = 3$) or Total Escape, ($\alpha = 4$) the machine discarded the card and read the next one in the stack. (AR:7)

The flow diagram also shows that census cards were discarded if the neutron had reached the end of the current census period. This procedure would have removed the need for any manual processing of the deck of cards until the end of a census period: processing an input deck of cards would produce an output deck which could be fed straight back into the machine. This process would terminate when an input deck gave rise to no new cards, indicating that all the surviving neutrons had reached the end of the census period. At this point, the complete set of cards would have to be sorted to separate out the census cards.

A possible explanation of this discrepancy is that the team began by feeding all the newly punched cards back into the reader for the next cycle, but then found that it was more efficient to sort the cards at the end of each cycle instead of waiting until the end of the census period. Reading and discarding punched cards would not have been an optimal use of ENIAC's extremely fast processing speed. This change in procedure would not have required any changes to the program, however, as the code for discarding cards or certain sorts would simply never get executed if the cards had already been removed from the input deck.

In either case, a new census period would begin with the census cards produced in the previous period. At this point, as described in Section 2.1 above, the cards would have required manual processing to restore the original sample size of 100, and then a new census period could be started if required.

> T, the census time interval was taken as 1 shake = 10^{-8}sec for all the problems in this run; the value of T was set by an external switch on the Constant Transmitter and manually increased by one at the end of each time-cycle. (AR:7)

2.7 Overview of the First run computation

Von Neumann's early Monte Carlo flow diagram was divided into a number of spatially disjoint regions connected by variable remote connections.[26] These regions corresponded to significant functional units in the program, and were later labeled and used as "black boxes" in a high-level sketch diagram that showed the main control flow through the program and helped structure the calculation of the number of loop iterations and hence the running time of the program. The description of the program in *Actual Running* is also based on these regions. This section uses this structure to give an overview of the First run calculation as shown on the December 1947 flow

[26] A notational device introduced in the first of the *Planning and Coding* reports.

diagram. The emphasis here is on the overall flow of the computation: section 5, below, gives more detailed descriptions of the evolution and internal structure of the individual regions.

The flow diagram begins with a box labeled 0* which makes an unconditional transfer to connector I. Box 0* is entered from a connector labeled Θ, which is not referred to elsewhere in the diagram. A possible explanation for this apparently pointless maneuver is that the connector Θ represents the first line in the function table, where execution would start, but that the code corresponding to connector I was placed elsewhere. The first instruction in the function table would then carry out an unconditional transfer to the location of I. This pattern is used in the Second run code, possibly to allow space for the insertion of unforeseen initialization code before the start of the program proper.

2.7.1 A_1 Read card and store neutron characteristics[27]

Boxes 1*, 1.1*, 1.2*, 1.3*, 2*, 2.1*, 3*, 5*, 7*, 8*.

This region begins by reading a card and examining the terminal event recorded on it (the value of α). If it is a "special" card, with $\alpha = 7$, the values of ξ and $\bar{n}$ it contains are stored, and the next card is read. Cards with $\alpha = 3$ or $\alpha = 4$, representing neutrons that have escaped or been absorbed, are also ignored and another card read.

If the card is a census card, with $\alpha = 5$, its time value t is stored and compared with the current census time T set on the constant transmitter. If the card's time is greater than the current census time, the card is ignored and another read.[28] Otherwise, the values of the neutron's position, direction and velocity are stored.

If the card is not a census card, the terminal event it records must be fission, and its "weight" q, denoting the number of new neutrons generated by the fission, is stored. The random number ξ is refreshed (i.e. a new value is calculated via a call to the random number subroutine) and this value is used to determine new randomly allocated values for the neutron's direction and velocity. These values are stored, as is the value of the neutron's position read off the card.

At this point (just before box 8*) the execution paths for census and fission cards merge, the time of the terminal event and the number of the neutron's current zone are read from the card and stored, and the computation proceeds to A_2.

Following box 8*, a storage table lists the five values that have been stored in the preceding section and the numbers of the accumulators in which they are stored.

2.7.2 A_2 Calculate random parameter λ^*

Boxes 1°, 2°, 3°, 4°, and two associated specification boxes.

This region calculates the random parameter λ^* which is used in region E to determine the expected distance to a collision.[29] This is stored in accumulator 4, and the computation proceeds to region B.

2.7.3 B Find neutron's velocity interval

Boxes $\overline{1}$, $\overline{2}$, $\overline{3}$, $\overline{4}$, $\overline{5}$, $\overline{6}$, $\overline{7}$.

---

[27] Region A carried out two distinct pieces of functionality which were separated and independently developed in the Second run program. We use subscripts to mark this distinction.

[28] The expression used to perform this test is $t - T + \tau_0$. The value τ_0 is not referred to elsewhere in the material we have examined, and its meaning and purpose is unclear.

[29] This parameter has the same role in the calculation as the parameter λ referred to in von Neumann's original plan, but was calculated in a different way.

This region locates the velocity interval containing the current neutron's velocity. A storage table following boxes $\overline{4}$ and $\overline{5}$ notes that the index k of the relevant interval is stored in accumulator 3; it will be used in region D to find the relevant cross-section values for the neutron. The computation then proceeds to region C.

2.7.4 C Calculate distance to zone boundary

Boxes 18.0*, 18*, 19*, 20*, 21*, 22*, 23*.

This region calculates how far the neutron will have to travel along its current trajectory before reaching a zone boundary. The results are recorded by the variables ε, representing the direction of travel (inward: $\varepsilon = 0$, outward: $\varepsilon = 2$), and d, the distance to the boundary of the neutron's current zone. The storage of these values is recorded in a storage table following box 23*, and the computation proceeds to region D.

2.7.5 D Calculate cross-section of material in zone

Boxes $\overline{14}$, $\overline{15}$, $\overline{16}$, $\overline{17}$, $\overline{17.1}$, specification box following $\overline{17.1}$, and 24*.[30]

This region calculates Σ, the total collision cross-section of all the material in the neutron's current zone. Boxes $\overline{14}$ to $\overline{17}$ form a loop which is executed once for each material type: b is initialized to 0 in box $\overline{14}$, tested in box $\overline{17}$, where the loop is exited if $b \geq 3$, and incremented in box $\overline{17.1}$. For each value of b, box $\overline{16}$ forms the sum of the $S_c(h)$ for the velocity interval index k calculated in region B ($h = 10b + k$). This value is then multiplied by f_i^b, and the product added to the running total, g, which in a substitution box following box $\overline{17}$ is defined to be Σ; a storage table following the specification box records that Σ is stored in accumulator 10.

In order to fit the tabulated values of the cross-sections into the available space, scaled values were stored on the numerical function table. In box 24*, the scaled total cross-section value Σ is adjusted to give the true total cross-section σ. The process of scaling is described in an informal annotation.[31] The computation then proceeds to region E.

2.7.6 E Determine if terminal event is collision, escape, or census

Boxes 25*, 26*, 27*, 28*, 29*, 29.1*, 29.2*, 30*, 30.1*, and specification box following 28*.[32]

In this region, the true cross-section value σ is used in conjunction with λ^* and d to determine whether the neutron will collide with another particle before it reaches the boundary of its current zone. If a collision takes place, box 26* sets d_1 equal to the distance to the collision and $\alpha^* = 5$ to indicate a collision. If no collision takes place, box 27* adjusts the value of λ^*,[33] and sets d_1 equal

---

[30] In the draft and sketch flow diagrams, box 24* is shown as belonging to region E. However, *Actual Running* discusses it along with the preceding boxes, and we have therefore chosen to include it in region D, which therefore contains all the operation boxes contributing to the calculation of the total cross-section value.

[31] In one copy of the December 1947 flow diagram that we have found, box 24* is crossed out and marked as "Changed". Furthermore, *Actual Running* notes that "operation boxes 24.0*, 24.1*, 24.2* and 24* … find and apply the appropriate correction factor" (AR: 11); however, the boxes 24.0*, 24.1* and 24.2* are not shown on the diagram. This suggests that there may be a later version of the First run flow diagram that has not been uncovered, and it is possible that these additional boxes formalized the calculation of the scaling factor shown in the informal annotation. The draft and December flow diagrams list four alternative scaling factors; perhaps the diagram was updated when the final choice was made of which factor to use.

[32] In the sketch flow diagram box 30* is shown as belonging to region F. However, it has a close functional relationship with region E, reflected in the fact that on the December 1947 flow diagram, the storage table recording the data calculated in region E is placed *after* box 30*; also, *Actual Running* discusses it in conjunction with the preceding boxes. We have therefore chosen to treat box 30* as part of region E.

[33] This adjustment is necessary for neutrons which move into an adjacent zone. As described in region F, the path of these neutrons is followed into the next zone before their fate is determined. This is done by continuing from region C

to d, the distance to the zone boundary, and $\alpha^* = 4$ to indicate a zonal escape. The two paths then merge. At this point, d_1 represents the distance the neutron travels before an escape or collision takes place.

It is now necessary to check if the neutron has time to reach this location before the next census point. Box 28* compares d_1 with the distance the unmolested neutron would have travelled by the end of the current census period. If d_1 is smaller, the specification box states that d_2 is set equal to d_1, and box 29.2* calculates t^*, the time at which the terminal event will occur. If d_1 is larger, however, the end of the census period is decreed to be the terminal event, and boxes 29* and 29.1* set α^* to 6, d_2 to the distance the neutron will have travelled by then, and t^* to T, the end of the census period.

At this point, the different paths of control again merge. d_2 holds the distance the neutron will travel before its terminal event occurs and t^* is the time at which that event will take place. Box 30* calculates r^* and a^*, the neutron's position and direction at t^*, and a storage table records the accumulators in which α^*, t^*, r^* and a^* are held. Box 30.1* moves ε (the direction of travel) and α^* to new locations,[34] and the computation proceeds to region F.

2.7.7 F Discriminate between different terminal events

Boxes 31*, 32*, 32.1*, 34*, 35*.

This region determines the subsequent fate of the neutron, depending on the class of terminal event identified in region E and coded in the variable α^*.

If $\alpha^* = 6$, the terminal event is census, and the computation continues at box 37* in region L to print the next card.

If $\alpha^* = 5$, the neutron has escaped from its current zone. The variable ε, recording whether the neutron is moving inwards or outwards through the assembly, is used to update the variable i holding the current zone number. If i is now greater than the maximum number of zones I in the assembly, the escape is total, and the computation path merges with that of the census neutrons in region L to print the next card.

If, on the other hand, $i \leq I$, the neutron has simply moved into an adjacent zone and its path can be followed further without printing a new card. In this case, known as "zonal escape", the neutron's characteristics are unchanged and the computation returns to region C.

Otherwise $\alpha^* = 4$, indicating that the neutron has suffered a collision. The computation proceeds to region G to calculate a new random number which will be used to determine the type of collision. Box 32.1* sets the "return address" so that the computation will then continue with region H.

2.7.8 G Refresh random number

Unnumbered box at bottom left margin of flow diagram.

---

without recalculating the value of λ^*, which is therefore here "reduced by the 'risk' $d\sigma$ that [the neutron has] already overcome, in order to keep the statistics unbiased" (AR: 11). For all other neutrons, the current value of λ^* is not reused.

[34] This box is not mentioned in *Actual Running*, raising the possibility that it was removed in a later version of the flow diagram. ε is moved to accumulator 12 which, according to box 1*, holds q, the weight of the current card; this may be an oversight, or it may reflect the practice visible in the Second run code of storing multiple short values in one accumulator. The "12" in box 30.1* is itself an amendment; the original assignment to accumulator 7 perhaps had to be changed because of its later use in box $\overline{18.1}$ to hold the value of f_i^b. α^* is moved to accumulator 1, from where it is punched onto a new card in box 47*.

This single box describes the calculation to refresh the random number ξ. It is entered by the connector ρ and left by the variable remote connector ω, which must be set by the caller before transferring control to ρ. This subroutine is called twice in the flow diagram, in region A_1 and between regions F and H.

2.7.9 H Determine collision type

Boxes $\overline{18}$, $\overline{18.1}$, $\overline{19}$, $\overline{20}$, $\overline{21}$, $\overline{22}$, $\overline{23}$, $\overline{24}$, $\overline{25}$, $\overline{26}$, $\overline{27}$.

This region determines which type of collision has taken place, and redirects the subsequent course of the computation accordingly. This process is controlled by a variable g, computed in box $\overline{18}$ to be the negative of a random fraction of the scaled total cross-section Σ.

In box $\overline{19}$, the product of the density of the first type of material in the current zone and the relevant partial cross-section for elastic scattering is added to g. If the sum is non-negative, it is decreed that the collision was elastic scattering; if not, the process is repeated, in boxes $\overline{21}$, $\overline{23}$ and $\overline{25}$, for inelastic scattering, fission and absorption. If the value of g is still less than zero, the whole process is repeated for the remaining material types.

The test in box $\overline{18.1}$ ensures that any material that does not occur in the current zone is ignored. The whole process is guaranteed to terminate because the total cross-section was calculated in region D to be precisely the sum of the partial cross-sections across all material and collision types.

2.7.10 J Elastic scattering

Boxes 51*, 51.1*, 52*, and specification box below 51*.

The effect of elastic scattering depends on the type of material the neutron collides with. It is assumed that if the collision is with a heavy nucleus, the neutron will simply bounce off it with the same velocity but in a new direction. If the collision occurred in light material, however, the velocity will change as well as some of the neutron's momentum is transferred to the light nucleus.

Box 51* checks whether the collision has taken place in light material, using the parameter μ_b. Box 52* determines the new velocity and direction in this case, using random digits specified in box 51.1*.[35] Otherwise, the specification box states that the velocity is unchanged, and box 53* (which is shared with the path for inelastic scattering) computes the new value of the neutron's direction.

The control paths then merge, and the computation proceeds to follow the history of this neutron further. Because a collision has taken place and the neutron's velocity may have changed, unlike in cases of zonal escape, the calculation in region A_1 of the parameter λ^* and in region B of the applicable velocity interval needs to be repeated, so the computation proceeds from region A_1.

2.7.11 K Inelastic scattering

Boxes 53*, 54*.

Inelastic scattering changes the neutron's velocity and direction. New randomized values for these are calculated and, as in the case of elastic scattering, the computation continues from region A_1.

2.7.12 L Absorption and fission

Boxes 36*, 37*, 37.1*, 38*, 39*, 46*.

---

[35] In fact, *Actual Running* states that none of the problems considered in the First run contained any light material. It is possible that a later version of the flow diagram removed the boxes dealing with light materials, as they are missing from diagrams prepared for the Second run.

In the case of fission, boxes 38* and 39* use random digits from ξ and the fission probabilities p_q^b to determine the number of daughter neutrons; this value is recorded as q^*, and box 46* increments the "fission generation" of the neutron

In the case of absorption, α^* is set to 3 and the control path merges with that coming from region F for total escape and census neutrons. q^* is set to 1 and the fission generation is copied from the input card but not changed.

2.7.13 M Print card and restart main loop

Boxes 37.1*, 47*, 48*, 49*, 50*.

For the terminal events of census, fission and absorption, a card is punched recording the new neutron characteristics. Box 37.1* ensures that all the data that needs to appear on the card is in the right place for printing, and box 46* does the printing, giving the complete layout of the cards for the First run.

In preparation for the next iteration of the main loop, box 48* then increments the card's serial number, $\bar{n}$. The value of q, the weight of the current card, is then examined. If it is 1, processing of this card is complete and the computation returns to region A_1 via the connector I to read a new card. If it is greater than 1, however, at least one of the daughter neutrons produced by the previous fission event has not yet been dealt with, and in this case the data current card needs to be run through the computation again. Box 50* decrements the value of q and the calculation returns to region A_1 via the connector K, where it joins the control path for fission cards.

3 The Second run calculation

Like the First run, the Second run calculation represented an incremental change over its predecessor rather than a completely new start. The physical model, the data held about neutrons, and the details of their lifecycles are broadly the same as in the First run, and the most significant changes were to do with the way the large-scale structure of the computation was organized.

The complete code for the Second run program is in the John von Neumann papers in the Library of Congress.[36] In addition, we found two flow diagrams: one is a complete diagram with numbered boxes, and the other appears to be an earlier draft. There is a very high degree of consistency between the code, the complete flow diagram, and the description of the Second run program in *Actual Running*.

3.1 Automating censuses

In the First run, card decks were repeatedly run through the machine until all the active neutrons had been followed as far as the current census time, which was manually set on the ENIAC's constant transmitter at the start of each run. The census time was then reset and, if necessary, the neutron population adjusted before processing continued. In the Second Run, steps were taken to automate this process.

> An attempt was made in the Second Monte Carlo run, to include in the logical sequence and coding of the program, an automatic way of handling the beginning and ending of a time cycle, i.e. the individual neutrons belonging to different time cycles could be calculated instead of having to wait until all surviving neutrons have been driven to the end of the same census interval. (AR: 5-6)

[36] 30 manuscript pages with cover sheet titled "Card Diagram, FLOW DIAGRAM, Coding, Function Table III Values, Monte Carlo Second Run", JvN-LoC, box 12, folder 5.

> [I]t was desired in this run to make, if possible continuous computation, without having to stop the machine in order to permit the manual resetting of the value of T at the end of each census cycle, (i.e. when all neutron cards still "alive" in the assembly, have reached time T), and to be able to feed cards back into the "Reader" of the machine without having to separate them, by outside sorting, into census stacks. (AR: 12)

One consequence of this was a change to the way that time was represented. Time was still thought of as divided into census intervals, but the current census interval would now be recorded on each card.

> On the other hand, it was desired, however, to note on each neutron card how many times it or its ancestors have passed through a census time during the course of the whole computation. In order to achieve these requirements, T, the census time unit, was replaced by t_i the integer part of t. It was increased by one each time the terminal event, of the neutron under consideration, was determined to be census while t_d, the decimal part of t, was treated in the same way as in the First run. It was used to keep track of the time changes within a census cycle. t was, however, used in the Second run as a negative number, i.e. starting the calculation of a problem at $t = -X.0000$, where X could be chosen according to how many census cycles were desired before t_i reaching 0, would stop the calculation. Census time was started with $t_d = 0000$ and ended when $t_d \geq .8192$. (AR: 12-13)

In other words, a neutron's start time was stored as a negative number, and when in the course of the computation this value reached zero, processing for that neutron was stopped.[37] At the end of a census interval, measures had to be taken to keep the neutron population large enough to ensure that a reasonable sample was obtained.

> With this method no attempt was made to keep the input stack at a fixed number, however, in order to insure a sufficiently great sample population at the end of the calculation, all neutrons surviving the end of a census interval, were given a double weight before printing, (i.e. $q^* = 2$), thereby doubling the surviving neutron population at the beginning of the next time cycle. (AR: 6)

3.2 Reusing "virgin" cards

A further, related change in the Second Run was to distinguish between two types of reaction: those that were "under-critical", where the neutron population decreased over time, and those that were critical or "super-critical", where the neutron population increased over time. In the first case, the initial card stack was repeatedly read, modeling a continuing source of new neutrons at the center of the assembly:

> The original input cards were also treated differently in the Second Monte Carlo run, further distinguishing between the under-critical and critical or near-critical assemblies. An attempt was made to make more efficient use of the machine computing time, i.e., to increase the sample population as the neutrons in the assembly approach, with time, the expected normal spatial and energy distribution. In the under-critical cases, assuming a neutron source at the center, the original or "virgin" input fission neutrons were again started at the center of the assembly, with a random fission velocity but spaced in time by equal intervals γ. At each census T, the value of γ was halved, thereby doubling the number of "virgin" cards to be read in the next time cycle. This had the intentional effect that half of all the "virgin" neutrons thus generated were read and their path computed in the last or latest census-cycle. (AR: 6)

In the critical and super-critical cases, however, the procedure was different:

> In the case of the critical or super-critical assemblies, a method similar to the First run was used, starting, however, with a small initial stack of about twenty neutrons at some original T, but effecting the increase of the sample population, (besides its natural increase), by doubling at the end of each census cycle as described above. (AR: 6)

---

[37] It is not clear from the Second run flow diagrams or code exactly how the calculation stopped when t reached 0.

3.3 Zonal escape

Certain properties of the physical nature of the problems considered in the Second run necessitated a change in the handling of zonal escape and in the way that zones were thought of.

> In the First run zone limits were chosen according to the changing material composition in the assembly. In some of the problems of the Second run, bands of the same material composition were split up into several zones in order to facilitate the calculations. (AR: 2).

> Some of the problems in this run were computations on spherical assemblies consisting of 2 or 3 material zones, the outer zone (tamper) having disproportionately large dimensions from the point of view of calculation, i.e. the statistical information obtained through these calculations might have been rendered unwieldy by the size of the outer region. Also, because of the very large scattering cross-sections in the tamper, relatively little information regarding other types of collision could have been obtained between census-cycles in this region. Therefore, for this type of problems, zones of the same material composition were divided into several zones of varying width and a weighting system was devised in the following manner: Two additional symbols for α^* were introduced to indicate the nature of the zonal escape, if after examining ε it was determined that the neutron particle was travelling "in" towards the center, its weight was doubled, ($q^* = 2$), and $\alpha^* = 7$ noted the event zonal escape. If, on the other hand, the neutron was found to be travelling "out" towards the surface of the assembly, a game of chance was played to determine whether or not after the present event the path of the neutron under consideration should be followed further. If the game was "won", α^* again became 7, but $q^* = 1$ – if the game was lost $\alpha^* = 2$, indicating a neutron whose genealogy was ended with this zonal escape and $q^* = 0$ for the same reasons as mentioned in the case of total escape. (AR: 16−17)

3.4 The physical model

The changes to the handling of zonal escape did not affect the nature of the data stored, however, and the physical model used in the Second Run was very similar to that of the First Run. However, the values of the cross-section functions were no longer tabulated, as they had been in the First Run:

> However, on Monte Carlo Flow Diagram II, this whole calculation [of the total collision cross-section of the material in the current zone] was included in one operation box (46), i.e. by "pre-mixing" through hand-calculation the cross-sections of all materials existing in the same zone, (we had allowed four materials in each zone of the First run) we could omit the separate calculations for each b in that zone; this arrangement, and some further simplifications to be explained later, released sufficient storage space on the numerical Function table to store the true total cross-sections with sufficient significant figures and already multiplied with the material density. (AR: 11)

As before, the numerical data describing a particular problem was placed on the third ("numerical") function table. The data stored was the following:

- $(\sigma_E/\sigma_t\,, \sigma_I/\sigma_t)_{bk}$: relative values of elastic and inelastic scattering σ_E/σ_t and σ_I/σ_t, for each pair of material b and velocity interval k were stored in the A field of rows 0 – 39.
- $\sigma(h)$, the values of the cross-section functions, were stored in the B field rows 0 – 39. The values of b (material type) and k (velocity interval index) were combined into a value $h = 10b + k$, where $0 \leq h \leq 39$. Then for each of the four values of c (process type), the values of $S_c(h)$ were stored in row h of the function table,
- V_k, the velocity interval boundaries for $0 \leq k \leq 9$, were stored in rows 40 – 45. Two values were held on each row, as $-V_k$, and the value of V_5 was repeated.
- v_{bk}: parameters used to select number of daughter neutrons after collision in material b and velocity interval k. Stored in the A field of rows 46 – 85.
- Ten values $V(\xi)$, the "centers of gravity" of the 10 velocity intervals, were stored in the B field of rows 46 – 55. These were used to allocate a random velocity, selected using a random digit ξ, to a neutron generated as the product of a fission event.

- Ten values $c(\xi)$ were stored in the B field of rows 56 – 65. These were the precomputed values of $\cos\left(\frac{\pi}{10}\left(\xi+\frac{1}{2}\right)\right)$, for random ξ, $0 \le \xi \le 9$, used in the calculation of a neutron's velocity after elastic scattering had occurred.
- Ten zone boundaries r_i, $0 \le i \le 9$, were stored in the B field of rows 66 – 75.
- The values of b_i, indices to the mixture of materials found in zone i, were stored in the B field of rows 76 – 85.
- σ_{2H}/σ_t, the relative cross-sections for elastic scattering in light materials, were stored in the B field of rows 88 – 97. There was only space in the function table to store the 10 values corresponding to the material in the first zone.
- A few constants used in the course of the program were also stored in the remaining space in the numeric function table.

3.5 Describing neutrons

In the Second run, the same basic terminal events were considered as in the First run, but there were some differences in the way the events were treated. In particular, the treatment of zonal escape was made more complex by the particular nature of some of the assemblies considered.

The mechanism of "special cards" was no longer used to reset the sequence of random numbers. The possible values of α in the Second run were therefore the following:

- $\alpha = 2$: Zonal escape, neutron not to be followed further;
- $\alpha = 3$: Absorption;[38]
- $\alpha = 4$: Total escape;
- $\alpha = 5$: Fission;
- $\alpha = 6$: Census;
- $\alpha = 7$: Zonal escape, neutron to be followed further.

A small number of changes were made to the data punched onto cards to describe neutrons. The "fission generation" and the serial number of the neutron's "ancestor" were no longer recorded:

> In view of the added need for storage space both on the IBM cards and on the machine and, since with the help of n', the "parent" number, it was possible to trace back neutron generations if this was necessary during the analysis of the results, g and for similar reasons n^*, the original "ancestor" numbers were both omitted in the Second run. (AR: 14)

Five new pieces of data were included for use in the analysis of output cards but which did not play a material part in the calculations, were added.

- i_l the zone where the previous census (of the neutron) occurred
- i_m the furthest zone from the center of the assembly ever reached by this neutron or its ancestors
- k^* the neutron's velocity interval at the last terminal event
- k_l the neutron's velocity interval at its last census
- s the number of scatterings in the life of a neutron

The actual storage of this data on punched cards is described in box 60 of the flow diagram, and is summarized in the table below. In describing the calculation, the flow diagram has to refer to both

[38] In fact, the Second run program does not set α^* to 3 in the case of absorption. This is treated as equivalent to fission with 0 daughter neutrons, and as the card weighting is then 0, it is not reprocessed.

the original and the newly calculated values of many variables, and uses asterisks to indicate the newly calculated values.

Field	Print	Columns	Read
1	t^*	1 – 7	t
2	α^*	11	α
3	q^*	12	q
4	$\underline{n}$	16 – 19	Not read
5	q	20	Not read
6	ξ	31 – 40	Not read
7	$\overline{n}$	41 – 45	Not read
8	s	46 – 50	Not read
9	v	51 – 55	v
10	n'	56 – 60	Not read
11	i^*	61	i
12	i_l	62	i_l
13	k_l	63	k_l
14	i_m^*	64	i_m
15	k^*	70	k
16	r^*	71–75	r
17	a^*	76–80	a

In general, the table shows that newly calculated values (with asterisks) are printed at the end of the processing of a card, and that these are then read in as initial values (without asterisks) when the card is next read.

3.6 Operating procedure

The Second run program could read cards of two distinct input formats. Initial cards, sometimes referred to as "virgin" or "source" cards, were read using the code in region A_0. "Normal" cards represented the outcome of a previous event in a neutron's life, as in the First run, and were read in region A_1. The choice of whether to branch to region A_0 or A_1 was made at two places in the code: at function table address 015, following the initialization code in region S, or at function table address 124, after box 62, where after completing the processing of an input card, a choice is made to restart the calculation at connector *A* or *I*. These are coded as unconditional transfers, however, not conditional transfers. A hand-written annotation in the code at address 124 reads:

> (A) Virgin cards ~~Line~~ arg 17
> (I) For all others ~~Line~~ 26,[39]

and a similar annotation at address 015 states that the unconditional transfer could go to address 016 or 026.[40] It appears, therefore, that the program could be run in one of two modes: if the addresses at locations 015 and 124 were set to 16 and 17 respectively, the input stack consisted of "virgin" cards read in region A_0, but if the addresses were set to 26, the input stack consisted of "non-virgin"

[39] 17 and 26 are the function table addresses (lines) corresponding to the connectors A and I respectively. "arg" is presumably an abbreviation for "argument", the technical ENIAC terminology for the two-digit number that specified which line in a function table to access.
[40] Entering region A_0 at address 017 skips the read instruction at address 016. In both A_0 and A_1, a single input card could give rise to multiple passes round the main loop, and new cards were only read when necessary.

cards, i.e. those describing neutrons whose history had already been tracked for at least one time-cycle.

A run started by reading a deck of virgin cards A_0. Once these had all been processed, the addresses in 015 and 124 would be changed to read normal cards in region A_1. These would be handled in much the same way as in the First run, except that it was no longer necessary to sort the newly punched cards to separate out the census cards. All cards punched would be fed back into ENIAC, and those that required no further processing would simply be ignored by the input routine. A decrease in the number of cards being produced was taken to indicate that the assembly being simulated was under-critical, and in this case the virgin cards would be reprocessed to increase the neutron population.

As this example illustrates, the Second run allowed a number of computational options to be decided at "set-up time" by manually setting switches on the function table holding the program code before execution started. This was a new programming practice, not used in the First run, which allowed greater flexibility in the use of the code. Another manual setting made before starting computation affected the detailed processing that would be carried out in cases of zonal escape. This is described in region F, below.

3.7 Overview of the Second run computation

The continuity between the First and Second runs means that the Second Run calculation can naturally be described using the functional regions defined for the First Run.

As well as detailed changes within certain of the regions, there are two new pieces of functionality: region A_0 is introduced to describe the reading of "virgin" cards, and region N describes the new extended processing of zonal escape.

The only other large-scale change was the reordering of the regions A_1, B and C:

> In the Second Run, i.e. Monte Carlo Flow Diagram II, the operations to find λ^*, k, ε and d were, except for some storage location assignment, identical with the procedures as described above [for the First run], however, the logical order in which these values were computed or found , was somewhat interchanged for the following reasons: By interchanging the order obtaining first ε and d, then k and lastly λ^*, we could in certain cases, shortcut and, thereby save machine computing time, in following the path of some neutrons. Thus, if $\alpha = 6$ or 7, indicating that the terminal event on the neutron card just read was census or zonal escape, V remains unchanged and its k can be directly read off the card. In the case of the undercritical assemblies, since all "source" cards (neutrons) were started at the center, by assigning the value of $d = r_1$, the total radius of the first zone, and $\varepsilon = 2$, the outward direction, we were able to omit in the initial calculation that part of the sequence where these quantities are being computed. Since the computation to obtain λ^* is common to all neutrons considered here therefore by leaving it last in the sequence in this part of the flow diagram, we could merge at this part of the calculation all branches previously separated by the value of α. (AR: 9-10)

3.7.1 S Initialization

Box 1.

The Second run computation starts at the connector S. The initial serial number $\overline{n}$ is transferred from FT 000.2–4 to accumulator 17, and the initial value ξ_0 of the random number is taken from register J of the constant transmitter, rotated left by one digit, and placed in accumulator 16. $\overline{n}$ and ξ_0 are set manually before the computation begins. The computation then continues with region A_0 if a deck of virgin cards is to be read, or A_1 otherwise; the choice between these is determined by the setting of FT 015.3.

3.7.2 A_0 Read a virgin card and generate new neutrons

Boxes 0, 2, 3, 4, 5, 6, 7.

A virgin card is read in box 0. These cards are used to generate "new" neutrons which start at the center of the assembly with a randomly chosen velocity. Each card generates a number of neutrons, equally spaced in time. A virgin card holds three pieces of data: the initial and final times T_0 and T_1 of the interval in which the neutrons will be started, and a parameter γ representing the time interval at which new neutrons are started off. In box 2, the start time of the current neutron is calculated, and $\underline{n}$ is incremented in box 3.

If the end time T_1 has been exceeded, box 7 resets $\underline{n}$ to 0 and a new card is read. Otherwise, box 5 sets the new neutron's characteristics (it's in the first zone, travelling outwards) and box 6 assigns it a randomly chosen velocity. The computation then proceeds to region B.

3.7.3 A_1 Read card and store neutron characteristics

Boxes 10, 11, 12, 13, 14, 15, 16.

An input card is read and its weight tested. If the weight is zero, the card is discarded and a new card read. If the weight is greater than zero, the terminal event recorded on the card will be census, fission, or zonal escape, and the test in box 11 discriminates between these possibilities.

For census and zonal escape cards, the neutron's position, direction, velocity and velocity interval are unchanged and in box 14 are simply read from the card. If the neutron is the product of a fission event, however, its velocity and direction are unknown and new values are generated randomly in box 12. For these cards, the velocity interval has to be recalculated in region B, but this can be skipped for the others. Boxes 13 and 15 set the appropriate value of the variable remote connector Θ for these two cases.

The control paths for all neutrons then merge before box 16, which copies the zone number and time from the input card for all neutrons, and the computation proceeds to region C.

3.7.4 C Calculate distance to zone boundary

Boxes 21, 22, 23, 24, 25, 26.

This region calculates how far the neutron will have to travel along its current trajectory before reaching a zone boundary. This calculation is not performed for "virgin" neutrons for which these values are already known. The results are recorded by the variables ε, representing the direction of travel (inward: $\varepsilon = 0$, outward: $\varepsilon = 2$), and d, the distance to the boundary of the neutron's current zone.

The computation then branches at the variable remote connector Θ. For fission and virgin neutrons follow the connector Θ_1, and census and zonal escape neutrons follow Θ_2 to region B. The two paths merge again at the end of region B.

3.7.5 B Find neutron's velocity interval

Boxes 30, 31, 32, 33, 34, 35, 36.

This region locates the velocity interval containing the current neutron's velocity. This calculation is performed for virgin neutrons and fission products; it is skipped for census and zonal escape neutrons, in which cases the neutron's velocity has not changed. The computation then proceeds to region A_2.

3.7.6 A_2 Calculate random parameter λ

Boxes 40, 41, 42, 43, 44, 45.

This region calculates the random parameter λ which is used in region E to determine the expected distance to a collision. This is stored in accumulator 4, and the computation proceeds to region D.

3.7.7 D Calculate cross-section of material in zone

Box 46.

In the Second Run, the total material cross-section values were pre-computed for each mixture of material and stored in the numeric function table. The index b_i for the current zone i is looked up, and then the total cross-section value $\sigma(h)$, where $h = 10b_i + k$, is retrieved. The computation then proceeds to region E.

3.7.8 E Determine if terminal event is collision, escape, or census

Boxes 47, 48, 49, 50, 51, 53, 54, 55º, 56'.

In this region, the total cross-section value σ is used in conjunction with λ and d to determine whether the neutron will collide with another particle before it reaches the boundary of its current zone. If it does, box 48 calculates the distance d_1 to the collision and sets $\alpha^* = 5$ to indicate a collision; if not, box 49 sets d_1 equal to d and sets $\alpha^* = 4$ to indicate a zonal escape. The two paths then merge; at this point, d_1 represents the distance the neutron must travel before reaching the location of its terminal event, whether escape or collision.

It is now necessary to check if the neutron has time to reach this location before the next census point. Box 50 compares d_1 with the distance the unmolested neutron would have travelled by the end of the current census period. If d_1 is smaller, box 51 sets d_2 is equal to d_1, and box 53 calculates t^*, the time at which the terminal event will occur. If d_1 is larger, however, the end of the census period is decreed to be the terminal event, and boxes 55º and 56´ set α^* to 6, d_2 to the distance the neutron will have travelled by then, and t^* to 8192, representing the end of the current census period.

At this point, the different paths of control again merge. d_2 now represents the distance the neutron must travel to its terminal event, and t^* the time at which that event will take place. Box 54 calculates r^* and a^*, the neutron's new position and direction, and the computation proceeds to region F.

3.7.9 F Discriminate between different terminal events

Boxes 55, 55*, 56, 57, 64.

This region determines the subsequent fate of the neutron, depending on the class of terminal event identified in region E and coded in the variable α^*.

Box 55 checks whether the terminal event is census ($\alpha^* = 6$); if so, box 64 sets the weighting for the output card to 2 and the computation proceeds to region M to print a card.

Box 55* checks if $\alpha^* = 5$, in which case the neutron has suffered a collision. A new random number is calculated to help determine the type of collision and the computation proceeds to region H.

Otherwise $\alpha^* = 4$, indicating that the neutron has escaped from its current zone. In box 56 the variable ε, recording whether the neutron is moving inwards or outwards through the assembly, is used to update the current zone, i. If i is now greater than the maximum number of zones I in the assembly, the escape is total, and the computation path merges with that of the census neutrons and proceeds to region M.

If, on the other hand, $i \leq I$, the neutron has simply moved into an adjacent zone. The computation proceeds to region N, which implements the new Second Run treatment of zonal escape.

3.7.10 G Refresh random number

Unnumbered box at bottom left margin of flow diagram.

This single box describes the calculation to refresh the random number ξ. It is entered by the connector ρ and left by the variable remote connector ω, which must be set by the caller before transferring control to ρ. This subroutine is called twice in the flow diagram, in region A_1 and between regions F and H.

3.7.11 H Determine collision type

Boxes 65, 66, 67, 68, 69.

This region determines which type of collision has taken place, and redirects the subsequent course of the computation accordingly. This process is controlled by a negative random number g, extracted from ξ in box 65. As in the First run, the collision type is determined by adding partial cross-section values until g becomes positive.

In box 66, the relative cross-section for elastic scattering is added to g. The result is tested in box 67, and if it is positive the computation proceeds to box 74 in region J.

If not, the relative cross-section for inelastic scattering is added in g in box 68 and tested in box 69. If it is positive, the computation proceeds to box 77 in region K.

In the Second run, elastic scattering in light material was treated as a separate collision path. It is not shown on the flow diagrams, but is present in the code as a "special process". If the neutron was in zone 1,[41] the relevant cross-section was added to g and tested. If it is positive, the computation proceeds to box 52* in region K.

Otherwise, the collision has resulted in either a fission event or the absorption of the neutron, and the computation proceeds to region L.

3.7.12 J & K Elastic and Inelastic Scattering

Boxes 74, 75, 76, 77, 78, 52*.

For elastic scattering, the neutron's velocity does not change, but in the case of inelastic scattering, a new velocity is calculated in box 77. The control paths for the two types of scattering then merge: a new value for the direction is calculated and the parameter s, which records the number of scatterings in a neutron's lifetime, is updated.

For elastic scattering in light materials, new values of velocity and direction (?) are calculated and control joins the path for inelastic scattering.[42]

In all cases of scattering, the computation then proceeds with region C. In the case of elastic scattering, the variable remote connector Θ is set to skip region B as in this case the velocity interval is unchanged.

3.7.13 L Absorption and fission

Boxes 70, 71, 72, 73.

In box 70, the value ν for the current mixture of material and velocity interval was looked up in the numeric function table. The first digit of ν gives the number of daughter neutrons resulting from the

---

[41] There was only room in the numeric function table to store 10 relative cross-section values for elastic scattering in light materials, i.e. only enough data to allow for this possibility in one zone.

[42] Because in these two cases the neutron's velocity has changed, unlike in the normal case of elastic scattering where it is assumed to be unchanged.

current collision, and this is recorded as q^*, the weight of the card to be printed. The remaining digits of ν are compared with some random digits to decide whether or not to increment q^*.

The computation then proceeds to box 58 in region M to print the card.

3.7.14 M Print card and restart main loop

Boxes 58, 58*, 58°, 58', 59, 60, 61, 62, 63.

The control paths for total escape, fission and absorption then merge with the path for census, the "spatial parameters" and the neutron's velocity are updated, and the serial number is copied to the position where it will become the parent number on the output card, which is then printed.

The serial number is then updated and the scattering number reset to zero. The current weighting is examined, and depending on its value it is decremented and the current card reprocessed, or a new card is read for the next cycle.

3.7.15 N Zonal escape

Boxes 79, 80, 81, 82, 83, 84, 85.

At this point, the computation could follow one of two paths, determined by the setting of FT 110.5. In the basic case, box 79 set the value of α^* to 7 and the weighting to 1. If, on the other hand, the current assembly required the special handling of zonal escape discussed above, box 82 checked the direction of the neutron's travel; if it was found to be travelling inwards, its weight was set to 2. If it was travelling outwards, however, the "game of chance" was played by testing in box 84 whether a random digit was greater than or less than 5. If the game was "won", α^* was set to 7 and q^* to 1; if the game was "lost", α^* was set to 2 and the weighting stayed at 0.

The various control paths came then came together at box 80 to update the value i_m recording the outermost zone reached by the neutron, if necessary, and the computation proceeded to region M.

4 Summary of the evolution of the Monte Carlo programs

The previous sections have described the three major versions of the Monte Carlo computation largely as three separate programs. However, there is a great commonality of structure between the three versions, and several sections of the program remain virtually unchanged from von Neumann's initial flow diagram to the Second run program.

The following table summarizes this common structure. For each of the major versions of the program, it lists which operations or flow diagram boxes correspond to each of the functional regions used in the description of the programs in previous sections. These regions were defined in an overview flow diagram which broke down the draft First run diagram into regions labelled A, B, C, D, E, F, G, H, J, K, L and M. Each region defines a more or less coherent part of the overall computation. This explicit referencing of regions was not retained for later flow diagrams, but provides a very useful way of structuring discussion of the programs, as the overall functional structure was largely preserved. In many cases, the boundaries between these units are marked on the December 1947 diagram by annotations noting the variables just calculated and the accumulator they have been assigned to.

Region	Description	Original plan	JvN Draft FD	First run	Second run
A	Read a card and store neutron characteristics	0	0* – 8*	1* – 8*	Restructured Virgin cards: 0 -6 Output cards: 10 – 16
	Calculate random parameter λ*		9* – 17*	New algorithm 1° – 4°	40 – 45
B	Find neutron's velocity interval		$\overline{1}$ – $\overline{13}$	Simplified $\overline{1}$ – $\overline{7}$	30 – 36
C	Calculate distance to zone boundary	1 – 15	18* – 23*	18* – 23*	20 – 26
D	Calculate cross-section of material in zone		$\overline{14}$ – $\overline{17.1}$	$\overline{14}$ – $\overline{17.1}$, 24*	46
E	Determine if terminal event is collision or escape	16 – 47	24* – 27*	25* – 27*	47 – 49
	Determine if a census comes first		28* – 29*	28* – 30*	50 – 54
F	Discriminate between terminal events	47	30* – 35*	31* – 35*	55 – 57
G	Refresh random number		Inline code 6*	Subroutine ρ/ω	Subroutine ρ/ω
H	Determine collision type	48 – 53	$\overline{18}$ – $\overline{27}$	$\overline{18}$ – $\overline{27}$	65 – 69
J	Elastic scattering	54 – 59, 61 – 68	51* – 52*	51* – 52*	74 – 76
K	Inelastic scattering		53* – 54*	53* – 54*	75 – 78
L	Absorption/fission	54 – 59, 65 – 81	36* – 46*	Simplified 36* – 39*, 46*	70 – 73
M	Print card and restart main loop	No looping 51, 69, 73, 77, 81	47* – 50*	37.1*, 47* – 50*	58 – 64
N	Zonal escape	48 – 50	Computation loops without printing		New process 79 – 85

Regions A – M were defined in an overview diagram that closely matched the draft flow diagram. The descriptions of the regions and the allocation of boxes to regions in MC1 and MC2 derive from the descriptions given in *Actual Running*. The very different style of the original computing plan means that the allocation of steps 1 – 81 to regions is in some places rather arbitrary.

The table makes clear that von Neumann's draft flow diagram employs two different box number schemes. It seems to have been produced in two stages: first, by implementing the original plan together with Richtmyer's suggestions of 2 April, 1947 (including census periods) (boxes 0* – 54*), and then later adding the sections that depended on the introduction of velocity intervals (boxes $\overline{1}$ – $\overline{27}$). We have not found any archival material predating the document describing the complete draft flow diagram, however.

The development of the Second run program introduced a systematic renumbering of the boxes to get a single numeric sequence. Note the reordering of regions A, B and C in this run. The Second run also introduced a new treatment of zonal escape, which is represented in this report by an additional region (N).

5 Details and evolution of functional regions

This section describes in more detail the computations performed in each of the functional regions, and highlights the differences, where they exist, between the three versions of the program.

5.1.1 S Initialization

First run: box 1.1*.
Second run: box 0.

Two pieces of data are provided at the beginning of a run: the initial value of the card serial number $\overline{n}$ to be used, and an initial value ξ_0 of the random number. In the First run, these values were supplied on special cards with $\alpha = 7$, but in the Second run they were set by hand on the function table and constant transmitter.

In the Second run code, the value ξ_0 read from the constant transmitter was rotated one digit to the left before being stored in an accumulator. The purpose of this is unclear, though it may be part of an attempt to use shift orders to "add to the randomness of the calculation" (AR:22). It is possible that the shift order in FT 014.4 was set manually before the computation restarted, to rotate ξ_0 by varying amounts and so "add to the randomness of the calculation", but we have no evidence to confirm this hypothesis. In the Second run, however, rotations of the digits of ξ were also added to the "refresh random number" subroutine: see Section G, below, for further discussion of this point.

5.1.2 A_0 Read a virgin card and generate new neutrons

Second run: boxes 0, 2, 3, 4, 5, 6, 7.

This region was introduced in the Second run. It reads data from a stack of "virgin" cards, whereas A_1 reads data from cards that hold details of an event in a neutron's life history. Each virgin card holds three values, T_0, T_1 and γ, and these are used to generate a sequence of neutrons starting at regularly spaced intervals of γ, starting at T_0 and ending at T_1.

In draft flow diagram for the Second run, this process is quite straightforward. The value of t, the start time of the neutron, is calculated as $t = \underline{n}\gamma + T_0$, where $\underline{n}$ is incremented each time a neutron is generated. The computation only proceeds if $T_1 - t \geq 0$; otherwise, t has gone past the terminal time T_1 and a new card needs to be read.

By the time the final flow diagram had been produced, however, the new representation of time described in section 3.1 above had been adopted, and time values were represented as seven-digit negative numbers of the form $t = -t_i.t_d$. Census intervals were divided into 8192 subintervals, and the four-digit decimal part t_d recorded the passage of time within a census interval. The three-digit integral part t_i indicated how many census periods the neutron had passed through. Annotations in the code listing show time values such as "M9891808000", corresponding to the number -010.8192.

This change had two consequences for the way the formulas in the flow diagram were written. Firstly, the simple variable t was replaced by $10^{-3}t$: as the decimal point in an accumulator was often assumed to be at the far left, the actual value stored could be thought of as $-0.0108192 = 10^{-3} \times -010.8192$. The calculation of the start time of a neutron was therefore expressed as $10^{-3}t^* = 10^{-3}(+T_0 + \underline{n}\gamma)$. The test of whether the terminal time had been reached was expressed simply as $t - T_1 \geq 0$, presumably because $t - T_1 \geq 0$ iff $10^{-3}(t - T_1) \geq 0$

An annotation on the flow diagram notes that the values read from the virgin card were $+T_0$ and $-T_1$; however, notes in the code listing give example values of M9801808 = –019.8192 for T_0 and 010.0000 for T_1, apparently reversing the specified signs of these values. To make sense of this, it is necessary to remember that time values were *negative* and to interpret the + and − signs on the flow diagram as operators rather than signs of magnitude. $-T_1$ therefore denotes a positive number and $+T_0$ a negative number, as required.

Actual Running notes that:

> At each census *T*, the value of γ was halved, thereby doubling the number of "virgin" cards to be read in the next time cycle. (AR: 6)

This does not happen in the code: presumably each virgin card in a deck represented one time cycle and was provided with its own value of γ. What would be doubled would be the number of initial neutrons generated, not the number of virgin cards.

5.1.3 A_1 Read card and store neutron characteristics

First run: boxes 1*, 1.1*, 1.2*, 1.3*, 2*, 2.1*, 3*, 5*, 7*, 8*.
Second run: boxes 10, 11, 12, 13, 14, 15, 16.

This processing in this region is essentially the same in all versions of the program, with the details changing according to the terminal events being dealt with and the way card weights are handled. In the First run, the value of α was used to decide whether or not a card needed to be processed; in the Second run, however, the card's weight q was used to determine this.

The basic function of the region is to correctly initialize the neutron's characteristic variables; in most cases these are copied from the card, but if the neutron is the product of a fission event its velocity and direction are unknown at this point, and new values are randomly assigned.

The second run flow diagram shows the read operation in an unnumbered box, and box 10 shows the assignment of q to accumulator 2 taking place before the conditional transfer. In the code, however, the assignment takes place after the transfer. The redrawn diagram therefore splits box 10 into three separate boxes containing the read operation, the conditional transfer, and the subsequent assignments, respectively.

5.1.4 A_2 Calculate random parameter λ*

First run: boxes 1°, 2°, 3°, 4°, and two associated specification boxes.
Second run: boxes 40, 41, 42, 43, 44, 45.

Randomization was used in the Monte Carlo programs to determine the point at which a neutron was deemed to have collided with an atomic nucleus. In the letter to Richtmyer, von Neumann explained the procedure as follows:

> The probability that the neutron will travel a distance d^1 without suffering a collision is 10^{-fd^1} … It is at this point that the statistical character of the method comes into evidence. In order to determine the actual fate of the neutron, one has to provide now the calculation with a value λ, belonging to a random variable, statistically equidistributed in the interval 0, 1 … Then it is decreed that 10^{-fd^1} has turned out to be λ; i.e.,
>
> $$d^1 = \frac{-^{10}\log\lambda}{f}$$

In other words, given a suitable random value λ, the distance the neutron would travel before a collision could be simply calculated from f, the sum of all the collision cross-sections in the current zone. Von Neumann originally proposed that a value of $-^{10}\log\lambda$ should be provided on each punched card, but in the First and Second runs this value was calculated when needed.

> Continuing the path of a neutron, next the number λ^* is obtained by another random process: It is the (negative) logarithm of a random number which is equidistributed between 0 and 1. λ^* is then stored in its assigned accumulator and later, in the course of the computation used to determine the place (distance) of the next collision: It will be used to define the product of that distance with the total collision cross-section of the zone in question. (AR: 8-9)

Von Neumann's original algorithm for this, given in the draft First run flow diagram, was soon replaced by a calculation which calculated λ^* using the approximation

$$\lambda^* = -\ln\xi = 2\left(\frac{\xi-1}{\xi+1} + \frac{1}{3}\left(\frac{\xi-1}{\xi+1}\right)^3 + \frac{1}{5}\left(\frac{\xi-1}{\xi+1}\right)^5\right),$$

as shown in box 3° of the First run flow diagram and box 45 of the Second run flow diagram (where the symbol λ^* is replaced by λ).[43] This formula was only applied for $0.5 \le \xi < 1$, however, so starting with an initial value ξ_0 derived from selected digits of ξ, $0 \le \xi_0 < 1$, a simple loop formed the sequence of values $\xi_{i+1} = 2\xi_i$ until a value n with $0.5 \le \xi_n < 1$ was found. $\alpha_n = n\alpha$, where $\alpha = \ln 0.5$, was then added to the calculated value of the logarithm of ξ_n to give the final value of λ^*.[44] This algorithm, and the flow diagram formulas used to express it, remained unchanged from this point through to the Second run program and beyond.

5.1.5 B Find neutron's velocity interval

First run: Boxes $\overline{1}$, $\overline{2}$, $\overline{3}$, $\overline{4}$, $\overline{5}$, $\overline{6}$, $\overline{7}$.
Second run: boxes 30, 31, 32, 33, 34, 35, 36.

This region locates the velocity interval, indexed by the variable k, containing the neutron's velocity v. Essentially the same algorithm was used for this throughout the development of the Monte Carlo programs, but the details of its implementation evolved significantly throughout the development of the First run program.

---

[43] The new algorithm is described on three manuscript pages which contain an explanation of the method, two flow diagram fragments expressing it in the style of the draft flow diagram, and some test calculations: JvN-LoC, box 11, folder 8. The simpler flow diagram fragment is essentially the same as the corresponding section in the December 1947 First run flow diagram.

[44] α here is a constant set on the numeric function table, and bears no relationship to the variables α and α^* used to record the types of terminal events.

> By using the given or, in the case of a fission neutron, the randomly chosen velocity, we then located the appropriate velocity interval, $0 \leq k \leq 9$ among the ten representative intervals. This was done by matching the V of the neutron to V_k, the values of the ten velocity interval limits which have been set on the numerical function table at the beginning of the run. (AR: 9)

A table on the page preceding von Neumann's draft flow diagram shows the original planned layout of the interval boundaries on the numeric function table (in face negative values of V_i were stored, to simplify the coding of the comparison against the positive value of v):

Line of Fc. T., No.:	40	41	42	43	44
Content:	V_5	(V_1, V_6)	(V_2, V_7)	(V_3, V_8)	(V_4, V_9)

The algorithm described in the draft flow diagram set a variable m to 44 or 41, depending whether v was greater or less than V_5. m was used to determine which row of the function table to access next, and it was incremented or decremented until the required interval was found. The final value of k was derived from m in a slightly complex way.

The archives contain a small flow diagram that records a change to this original algorithm in which k is used instead of m as the variable that controls the loops.[45] Depending on the result of the comparison with V_5, k is initially set to 0 or 9, and the function table is read from row 41 on until the required interval is found. This means that m can simply be incremented at each iteration of the loop, but requires that the pairs of values V_6 and V_9, and V_7 and V_8, are swapped round in the function table; this change is recorded in the later summary of the layout of the numeric function table in the First run.

As well as leaving the search loop when the appropriate interval boundary was located, both these versions of the algorithm checked explicitly for the situation where the loop index variable was about to go out of range. A further modification, shown in the December 1947 flow diagram, simplified this by noticing that this check could be removed if V_5 was repeated in row 45, in which case the loop would automatically terminate in all cases when the required value of k was found. The numeric function table layout shows the insertion of this addition line of data in the function table.[46] This version of the algorithm was used unchanged in the Second run flow.

> Having determined k, it is stored in its assigned memory location to be used later to find the appropriate cross-section of that velocity interval within the material zone in which the neutron under consideration is now travelling. (AR: 9)

Actually, as annotations in the flow diagrams point out, k is already safely stored at this point as a byproduct of the search algorithm.

5.1.6 C Calculate distance to zone boundary

First run: boxes 18.0*, 18*, 19*, 20*, 21*, 22*, 23*.
Second run: boxes 21, 22, 23, 24, 25, 26.

The algorithm used in this section remained essentially unchanged from the initial computing plan through to the Second run calculations.

> By using a, r and r_i, (r_i being the radii of the zones whose values were set on the numerical function table), we also determined the direction, ε of the neutron path, i.e. whether the particle was going in towards the center

[45] Undated manuscript page showing flow diagram with 9 boxes, some labelled $\overline{1}$ to $\overline{7}$: JvN-LoC, folder 11, box 8.
[46] We have found no evidence to indicate whether this change was made simply as a stylistic improvement to the algorithm, or out of a need to reduce the storage requirements of the program code (at the cost of an extra row of data in the numeric function table).

($\varepsilon = 0$) or out towards the surface of the spherical assembly ($\varepsilon = 2$), and the distance, d, that it can travel from its present position, in its present direction before reaching the next zone boundary. (AR: 9)

The calculation is slightly complicated by the fact that a neutron that is initially travelling inwards may end up travelling outwards and so move into an outer zone if its trajectory is such that it does not cross the inner boundary of its current zone.

In von Neumann's computing plan and the draft First run flow diagram, ε was set to –1 or 1, meaning that the sign of the intermediate value Δ could be altered by multiplying it by ε. The change to using 0 and 2 meant that this had to be carried out explicitly, but on the other hand the value of ε could now be stored in a single accumulator position without making use of the sign digit.

5.1.7 D Calculate cross-section of material in zone

First run: boxes ~~14~~, ~~15~~, ~~16~~, ~~17~~, ~~17.1~~, specification box following ~~17.1~~, and 24*.
Second run: box 46.

The different versions of the program differ considerably in the way they store the values of the collision cross-section functions, and consequently in the way that they calculate the value of the total cross-section which is used to determine whether or not a neutron suffers a collision.

In the Richtmyer plan, the cross-sections are given by the seven functions $\Sigma_{aA}(v)$, $\Sigma_{aT}(v)$, $\Sigma_{aS}(v)$, $\Sigma_{sA}(v)$, $\Sigma_{sT}(v)$, $\Sigma_{sS}(v)$, $\Sigma_{fA}^{(2)}(v)$, $\Sigma_{fA}^{(3)}(v)$, and $\Sigma_{fA}^{(4)}(v)$, and the total cross-section by the formula

$$f = \left(\Sigma_{aA}(v) + \Sigma_{sA}(v) + \Sigma_{fA}^{(2)}(v) + \Sigma_{fA}^{(3)}(v) + \Sigma_{fA}^{(4)}(v)\right)x_i + (\Sigma_{aT}(v) + \Sigma_{sT}(v))y_i + (\Sigma_{aS}(v) + \Sigma_{sS}(v))z_i$$

where x_i, y_i and z_i are the "relative volume fractions" of the materials A, T and S, respectively, in the current zone i.

Von Neumann noted that the Σ functions had a continuous domain (the velocity v could take any value within the fission spectrum), but made no commitment as to whether the required values would be found by tabulation, interpolation, or by a polynomial approximation. As described above, in the First run, the decision was taken to tabulate function values $S_c(10b + k)$ for the four process types c, the four material types b, and ten velocity intervals k.

In addition, the material density functions f_i^b were tabulated, as von Neumann had suggested, for the four material types b and ten zones i. If material b did not occur in zone i, the value of f_i^b should be zero. However, because the conditional transfer operation discriminated between values $<$ and $\geq$ zero, a slightly different approach made the details of the coding easier.

> f_i^b was set with a negative value, (-1) on the Numerical Function Table whenever the corresponding material was not present in the zone under consideration. The sign discrimination on f_i^b causes, for a negative f_i^b , the omission of the detailed calculation, as described above, for this material and immediately the next material will be examined. (N.B. similar procedure has been already used earlier in the calculations at the point where the sun of the total cross-section in a particular zone was being formed; see: Monte Carlo Flow Diagram I, operation box 15.) (AR: 18)

The calculation then proceeded as follows:

> As the next step in the calculation, we found the material or materials with their appropriate densities, as present in the zone; f_i^b the density of material b in zone i. The total collision cross-reaction of the material present at the previously determined k value was then formed, multiplied by f_i^b the material density of the zone. (AR: 10)

This approach meant that 160 cross-section values and 40 material density values had to be stored. The cross-section values were stored to three significant figures, which meant that four values could be stored in each row of the numeric function table. These occupied the first 40 rows of the table, and were indexed by the value of $h = 10b + k$. The 40 density values were held the left-hand fields of rows 46 to 85, again indexed by the value of h. Tabulating these functions therefore took up approximately 60% of the total space available in the numeric function table. It appears that this formed a significant constraint on the programming:

> For certain considerations of storage space on the numerical Function Table and, in order to be able to carry sufficient significant figures, scaled numbers S_J for the partial and Σ for the total cross-section, were used here instead of the true cross-section values. After having formed Σ it was multiplied by the appropriate scaling factor G in this manner obtaining σ, the true total cross-section. (AR: 10)

Perhaps for this reason, the approach taken to the calculation of the total cross-section value changed significantly in the Second run.

> However, on Monte Carlo Flow Diagram II, this whole calculation was included in one operation box (46), i.e. by "pre-mixing" through hand-calculation the cross-section of all materials existing in the same zone, (we had allowed for four materials in each zone of the First Run) we could omit the separate calculations for each b in that zone; this arrangement, and some further simplifications to be explained later, released sufficient space on the numerical Function table to store the true total cross-sections with sufficient significant numbers and already multiplied with the material density. (AR: 11)

Operation box 46 of the Second run flow diagram uses the zone number i to look up a single value b_i in the numeric function table; this is an index to the mixture of materials found in that zone. Using as before the value $h = 10bi + k$ as an index, the "pre-mixed" value σ_h was retrieved. 40 values of σ were stored in the right-hand fields of the first 40 rows of the numeric function table, meaning that the values could be stored to six significant figures. However, the saving in function table space was not quite as dramatic as these figures suggest: the left-hand fields of those rows were needed to store the cross-section ratios needed later in the computation to determine when elastic and inelastic scattering had taken place. In the First run, the stored partial cross-section values were used for both purposes. However, it was no longer necessary to store the material density values, so the strategy of "pre-mixing" reduced the storage requirements of the cross-section calculation by 50%.[47]

5.1.8 E Determine if terminal event is collision, escape, or census

First run: boxes 25*, 26*, 27*, 28*, 29*, 29.1*, 29.2*, 30*, 30.1*.
Second run: boxes 47, 48, 49, 50, 51, 53, 54, 55º, 56'.

At this point, the Richtmyer plan uses the values of λ (read from the card) and the total collision cross-section value f to calculate in steps 45 and 46 the distance d^1 that the neutron will travel before a collision.

> … it is decreed that 10^{-fd^1} has turned out to be λ; i.e., $d^1 = \frac{-10 \log \lambda}{f}$

In step 47, d^1 is compared with d, the distance to the zone boundary, to determine whether the neutron escapes from its current zone or suffers a collision before reaching the boundary.

[47] This freed-up space was used to store the values ν_{bk} used to determine the number of daughter neutrons emerging from a fission event, as described below.

In the First and Second runs the same procedure was carried out, though slightly differently expressed:

> The true total cross-section, σ times the distance d was then compared with λ^*, the quantity that was introduced above. If $d\sigma > \lambda^*$, we assumed, that at the point $d_1 = \lambda^*/\sigma$ a collision occurred, which was temporarily treated as a fission, ($\alpha = 5$). If, however, $\lambda^* > d\sigma$, the neutron was presumed to have escaped (temporarily noted as Total escape, $\alpha^* = 4$). Accordingly, in this case d was not modified, i.e. $d_1 = d$. (AR: 11)

In the First run, the control path processing neutrons which escaped into a neighboring zone rejoined the main path of the computation after the calculation of λ^*. In these cases, therefore, an adjustment was made to the value of λ^*.

> For the neutron path in the next zone the value of λ^* had to be reduced by the "risk" dσ that it had already overcome, in order to keep the statistics unbiased. This means replacing λ^* by $\lambda^{**} = \lambda^* - d\sigma$. (AR: 11)

(In the Second run, this was not necessary, as the organization of the computation ensured that a new value of λ was calculated for all neutrons.)

At this point, the computation had discriminated between the outcomes of zonal escape and collision. With the introduction of census periods in the First run, an additional check now became necessary. If the terminal event took place after the next census time, the terminal event for that neutron should be recorded as "census" rather than collision or escape.

> Next we examined whether there was indeed time enough (until the next census), for the neutron to cover (with its known velocity) the distance d_1, and thus really reach the collision or, in the case of escape, the limit of the zone. If covering the distance d_1 exceeds the census time T, the distance d_2 was formed for the point where the neutron arrived at census time, $d_2 = (T - t)v$, and the time of the event occurring was put equal to the census time, $t = T$ and the terminal event was noted as census, $\alpha^* = 6$. If, however, while covering the distance d_1, t failed to exceed T, a new value $t^* = t + d_2/v$ was formed, where $d_1 = d_2$. Merging the logical path for all neutrons, having suffered a collision, escape or census, we compute at this point the new values for a^* and r^*, since obviously any of the events will change the values of these quantities. (AR: 11–12)

In the Second run, the calculations in this section were essentially identical to those of the First run, apart from some differences arising from the different representation of time adopted to help automate the handling of census periods.

5.1.9 F Discriminate between different terminal events

First run: boxes 31*, 32*, 32.1*, 34*, 35*.
Second run: boxes 55, 55*, 56, 57, 64.

By this point a preliminary identification of the neutron's fate has been made, and the computation branches to handle each outcome in the appropriate way. In the First and Second runs, the census neutrons are handled first.

> Continuing now the course of the computation, we then examined α^* again to determine which path the particle will follow. If α^* was census, i.e. $\alpha^* = 6$, we gave q^* the appropriate weight: $q^* = 1$ in the First run where we manually duplicated or reduced each census stack to keep it an even hundred at the beginning of each census cycle, and $q^* = 2$ in the Second run where we automatically doubled the weight of each neutron which "survived" to the end of a census. g the fission generation, remained in this case unchanged from its previous value, i.e. since no fission occurred, the neutron was still within the same generation at which the particular card had started at the beginning of this computation. (In view of the added need for storage space both on the IBM cards and on the machine and, since with the help of n′, the "parent" number, it was possible to trace back neutron generations if this was necessary during the analysis of the results, g and for similar reasons n^*, the original ancestor numbers were both omitted in the Second run.) (AR: 13–14)

It then remained to discriminate between those neutrons which have escaped from their zone, and those which have suffered a collision. In the Richtmyer plan, a card was immediately printed for the escapees, and further processing carried out for the others to determine the type of collision. The same approach was taken in the First and Second runs for total escape:

> If the event was escape, $\alpha^* = 4$, we examined with the help of i, the present zone, ε, the direction of travel (in or out) and I, the total number of zones of the assembly under consideration, whether it was zonal or total escape. Total escape causes the same procedure as a census, i.e. it joins at this point the path preliminary to printing as described above. In the Second run, by putting $q^* = 0$ in the case of Total escape, an automatic removal from the Input stack, for those cards which should not be read again, has been facilitated. (AR: 15)

In the First Run, census and total escape cards were handled in exactly the same way, and the escape cards were manually removed before the new cards were fed back into the computation. In the Second run, this was performed automatically, the program ignoring any card whose weight was set to zero.

In the First run, however, no card was printed for zonal escape, and the neutron's current path was simply followed further. In the Second run, however, as discussed previously, cards were again printed for zonal escape neutrons.

> The event of zonal escape was treated in the First and Second Run considerably differently: In the First Run no card was printed if zonal escape occurred, but, after inspecting ε, thus determining and noting whether the neutron passed into the inner or into the outer neighboring zone, the computation was shifted back, with the help of an unconditional transfer order, to the point in the sequence (box 18.0*) where the direction and distance are being determined. Since no collision occurred, v remains unchanged and the total cross-section for the material in this zone is found by using the same k as in the previous calculation. λ^* has already been adjusted as noted earlier. The computation is then carried out as previously described until the next event has been determined and the point of the corresponding discrimination reached. (AR: 15–16)

In the First and Second runs, before proceeding to deal with the neutrons which had suffered a collision, a new random number was generated for use in determining the collision type.

5.1.10 G Refresh random number

First and Second runs: unnumbered box at bottom left margin of flow diagram.

In the Richtmyer plan, the necessary random numbers were to be provided on the input cards. It appears that a decision was taken early in the development of the Monte Carlo programs to have the ENIAC compute its own sequence of pseudo-random numbers, using the "square and take the middle digits" algorithm.

> Although lists of the checked random numbers had been available at the start of either run, it was found more convenient and faster to have the squaring, i.e. "refresh" done by the ENIAC. (AR: 22)

The draft First run flow diagram indicates the generation of a new number when required with the notation "$\xi^2 m$ to A". This appears four times: in box 6* to generate the velocity and direction of a fission product, in boxes 10* and 12* in the course of the computation of λ, and in box 33* to determine the kind of collision that has taken place.

The revised algorithm for the calculation of λ^*[48] reused digits from the number obtained in box 6*, getting rid of boxes 10* and 12* and leaving just two places in the computation where the random number was refreshed, namely boxes 6* and 33*. A one-page sketch flow diagram produced at this period shows operation box 6* in its normal sequence. Box 33* is missing, however; instead, an

[48] See discussion of region A_2, above.

arrow leads from box 32* to box 6* at the other side of the diagram, and a second arrow is added to the arrow leaving box 6* for box $\overline{18}$, thus informally showing two different successors to box 6*.[49] In the subsequent overview diagram that identified the different functional regions in the computation, box 6* is placed in its own region, G. This is placed between regions F and H but, confusingly, box 6* also appears in its normal sequence in region A.

A stable approach to representing this situation first appeared in the December 1947 flow diagram. On this diagram an explicit algorithm for refreshing ξ is given in a separate operation box which, by using the variable remote connection device to hold the "return address", is called as a subroutine at two places in the program.

> The "refresh" random number is shown on a separate block on the flow diagram, with ρ indicating the places where a new random number was needed and ω_n the various exits to the particular path where the computation had to be continued after obtaining the new random number. (AR: 22)

The connector ρ appears twice, at places corresponding to the original locations of boxes 6* and 33*. We have found a single sheet showing the coding of this refresh routine in the First run.[50]

This approach was carried forward to the Second run, and essentially the same algorithm was used in both runs, though there were differences in the way that the overall process was managed.

> A randomly chosen n digit number ξ was taken as the base ($n = 8$ in the First Run, $n = 10$ in the Second run), squared and the middle n digits used as, on the one hand, the next supply of random digits, as needed in the course of the computation and, on the other hand, as the base of the next number to be squared. This "refresh" method could have been repeated as many times as the calculations in the main part of the logical sequence required it. Since, however, in the first case, only two thousand and in the second case, little over three thousand squarings were statistically tested for randomness, provisions were made to recommence with the original random numbers each time when, in the case of the calculations two (three) thousand squarings have already been made. Having reached this point, the machine was instructed to read a "special" card, containing the original random number and to place it in the accumulator assigned as, more or less, permanent memory location for the changing values of ξ. This "special" card, as used in the First Run, also carried $\alpha = 7$ and $\overline{n}$, the serial number; $\alpha = 7$ was needed to separate the path of this card from those which were normally used in the course of the computation, $\overline{n}$ was used to retain intact the neutron card count. In the Second run, a count order, devised specially for this purpose, instructed the machine to stop after the appropriate number of squarings. Then ξ was replaced in its permanent memory location by the original random number changing, however, by appropriate shift orders, the order in which the digits were used to determine the various random paths for the neutron to travel. (This method was used in order to add to the randomness of the calculation.) (AR: 21–22)

It is not clear from the description above exactly what happened when the machine was restarted after 3000 squarings. The Second run code uses a shift operation to rotate ξ one digit to the left in between reading it from the constant transmitter and storing it in accumulator 16, an operation denoted on the flow diagram by the formula "$\xi_0 = \Theta\xi$". Assuming that the "original random number" is the one stored on the read-only constant transmitter, and that ξ's "permanent memory location" refers to accumulator 16, this makes sense of the description if we assume that the actual shift order executed was changed on the function table before restarting the machine. If not, there would be little point in doing it, as the same sequence of digits would be used each time.

In the Second run, the formulas "$\xi = \Theta^{-}\xi$'" and "$\xi = \Theta\xi$'" appear at the beginning and end of the operation box showing the refresh operation. The code corresponding to "$\xi = \Theta^{-}\xi$'" uses the S′R1 shift operation (FT171.3) to rotate ξ one place to the right. At the corresponding place in the code

---

[49] The way in which this is notated strongly suggests that the possibility of reusing box 6* only occurred to the author of the diagram as it was being written down.

[50] This is the only substantial bit of First run code that we have uncovered. Apart from the difference in algorithm, it uses a slightly different order code from the Second run program.

for "$\xi = \Theta\xi$'" (FT176.1), however, although illustrative annotations show ξ being rotated one place to the left, the shift operation is denoted by "S—", and the specific operation code 66 for the implied S′L1 shift operation has only been subsequently added in red pencil.

This provides some support for the hypothesis that the shift operations corresponding to the Θ and Θ^- operators were not fixed, but unfortunately we do not have enough evidence to prove this conclusively or to determine exactly what was done.

5.1.11 H Determine collision type

First run: boxes 18, 18.1, 19, 20, 21, 22, 23, 24, 25, 26, 27.
Second run: boxes 65, 66, 67, 68, 69.

In all versions of the calculation, a random number was used in conjunction with the collision cross-section values to determine the kind of collision that had. In the Richtmyer plan, the partial cross-sections f_i had been calculated in the course of determining f, the total cross-section.

> We can therefore now determine the character of the collision by a statistical procedure like the preceding ones: Provide the calculation with a value μ belonging to a random variable, statistically equidistributed in the interval 0, 1. Form $\overline{\mu} = \mu f$, this is then equidistributed in the interval 0, f. Let the seven above cases correspond to the seven intervals 0, f_1; f_1, f_2; f_2, f_3;, f_3, f_4; f_4, f_5; f_5, f_6; f_6, f, respectively. Rule, that that one of those seven cases holds in whose interval $\overline{\mu}$ actually turns out to be.

This process was carried out in steps 52 and 53 of the computing sheet and seven predicates Q_1 to Q_7 were used to record which kind of collision had been "detected".

Essentially this same approach was taken in the First run, with the difference that instead of seven alternatives (absorption, scattering by material A, T or S, fission producing 2, 3 or 4 neutrons), only four were considered (elastic and inelastic scattering, absorption and fission).

> Next, we follow the path were [sic] $\alpha^* = 5$, which indicates that a collision had occurred. We determine by a new random procedure what the specific character of the collision suffered by the neutron under consideration should be. This is done in the following manner.
>
> The scaled total cross-section, Σ is taken negatively and multiplied by a random number equidistributed between 0 and 1. The result is g. To this we add S_0, the scaled cross-section of elastic scattering, multiplied with the density, f_i^b of the first material in this zone. (The values for f_i^b have been set up on the Numeric Function Table in ascending order of b for each zone.) The result g_1 is then the first alternative of the four collisions to be considered. (AR: 17)

A loop is introduced here to consider each type of material in turn; if the value of f_i^b is negative, indicating that material b does not occur in zone i, the computation proceeds immediately to the next material type. This is the same control structure as used in region D to work out Σ. In both cases, for each material, it is necessary to consider the cross-section for each collision type: in region D, this is specified using a summation formula within an operation box, and the actual algorithm used to perform this "inner loop" is not made explicit. In this region, the four alternatives are considered individually, one after the other.[51]

> If g_1 is positive, that is, if an "overdraft" has already occurred, then this collision is decreed to have been elastic scattering. If, on the other hand, g_1 is still negative, then the procedure is repeated with the other successive alternatives in this order: Inelastic scattering, fission, absorption. The corresponding scaled partial cross-sections are S_1, S_2, S_3 and the successive results are g_2, g_3, g_4. If g_4 is still negative, this indicates that the first material was not responsible for the collision. In this case the procedure is repeated with the second,

[51] The fact that the central control code only allowed one "future control address" to be stored would have made it difficult to program nested loops, or a loop which branched to a different location on each iteration.

> third and fourth material, (four being the maximum number of materials allotted to each zone in these computations), until an "overdraft" occurs, i.e. until the random decision for the type of collision has been obtained. (AR: 17–18)

The changes in the representation of the cross-section values in the Second run necessitated some changes to this part of the computation.

> In Monte Carlo Flow Diagram II some changes, i.e. simplifications, have been made in this section of the calculation, retaining, however, the basis method of the random decision to determine the type of collision occurred. As mentioned earlier in this description, for the Second Run various materials existing in the same zone were "pre-mixed" and set on the Numerical Function Table already weighted by the appropriate material densities. This permitted then the omission of the use of f_i^b in the machine calculation and consequently shortened the calculation by having had to examine only one material for each zone in order to determine the collision path. Similarly, in the Second Run, fission and absorption cross-sections have been also "pre-mixed" by manual calculations with the effect that only two tries had to be made to determine which type of collision has been decreed. The relative values of Elastic and Inelastic scattering, $(\sigma_E/\sigma_t\,, \sigma_I/\sigma_t)$ are set on the Numerical Function Table. If after adding them to the negative random number, no "overdraft" occurred, no further try was necessary and the calculation was then made to follow the fission-absorption path. (AR: 18–19)

In the First run, elastic scattering in light material was treated as a special case of elastic scattering (see below). However:

> In the Second run, Elastic scattering in light material was treated as an added, separate collision-path, g_3 computing a new value of v for the elastically-scattered particle only if the "overdraft" occurred at this point. (Not shown on Monte Carlo Flow Diagram II.) (AR: 19)

To compute g_3, another relative value of the cross-section (σ_{2H}/σ_t) had to be added to g_2. However, there was only room in the numeric function table to store these values for one of the material zones. If the "special process" was invoked, therefore, the first thing to be checked was the current zone number: if the neutron was not in zone 1, the computation proceeded straight to the fission-absorption path.

5.1.12 J & K Elastic and Inelastic Scattering

First run: boxes 51*, 51.1*, 52*, and specification box below 51*; 53*, 54*.
Second run: boxes 74, 75, 76, 77, 78, 52*.

If a neutron is scattered, its direction and possibly its velocity are replaced by new, randomized values. In the Richtmyer plan, a new value for the velocity was calculated for the appropriate material type A, T or S.

> Scattering by A: Provide the calculation with a value ν belonging to a random variable, statistically equidistributed in the interval 0, 1. Replace v by $v' = v\varphi_A(\nu)$.

Scattering by T or S was handled in the same way. The direction was calculated as follows:

> As pointed out before, the "new" s will be equidistributed in the interval $-r^*, r^*$. It is therefore only necessary to provide the calculation with a further value ρ', belonging to a random variable, statistically equidistributed in the interval 0, 1. Then one can rule that s has the value $s' = r^*(2\rho' - 1)$.

In the First run, an explicit distinction was made between elastic and inelastic scattering. The clearest description of how this worked can be found in the draft document entitled "Actual Technique".

> Following the path of the positive branch of g_1 – the path the neutron would travel, if by the above-described random method the collision was decreed to be elastic scattering, provisions were made to compute a new value for $\underline{V}$ and $\underline{a}$, this however, would have to be computed only in those problems where any light natural

> was present in the zone. If, as in the six problems under discussion here, no light material occurred V remained unchanged, merging with the path of inelastic scattering after V has been recomputed on that branch, i.e. the positive branch of the g_2 sign discrimination. (AT)

This suggests that in the First run, the calculations described in box 52* were never carried out. We don't know whether the First run code included these calculations, but the Second run flow diagrams omitted the code for this case. However, as noted above, the Second run code did deal with elastic scattering in light material.

> If the "overdraft" occurred at g_2 i.e., the collision was decreed to be Inelastic scattering, a new value of v was computed and the path merged with that of Elastic scattering at the point where the computation to obtain the new value of a commenced. After this operation, an added step was done in the Second Run in order to increase by one the count s representing the number of scatterings suffered by the card, (neutron) under consideration. (AR: 19–20)

This is done in all cases of scattering.

> Then, through the means of an unconditional transfer, the computation was shifted back to that part of the logical sequence where the calculations for the new values of ε, α, λ and, if necessary, k are effected. The calculation is then followed on its path until an event other than scattering, i.e. census, escape or fission-absorption, has occurred. If the event is census or escape, the corresponding path, (described on the previous pages), will be followed in the computation. (AR: 19–20)

In other words, scattering is not a terminal event: the neutron's velocity and direction might change, but once these are adjusted, we can continue to observe its progress until something more dramatic happens. In the Second run, the address θ is set to θ_1 in the case of elastic scattering, and θ_2 for inelastic scattering. This means that in the case of elastic scattering, when the neutron's velocity is unchanged,, region B, which calculates the velocity interval, is skipped the second time the neutron's path is followed.

5.1.13 L Absorption and fission

First run: boxes 36*, 37*, 37.1*, 38*, 39*, 46*.
Second run: boxes 70, 71, 72, 73.

In the Richtmyer plan, the terminal events of absorption and fission with 2, 3 or 4 daughter neutrons are defined by separate cross-sections, and identified in the calculation by distinct predicates Q_1, Q_5, Q_6 and Q_7, respectively. For absorption,

> [t]he neutron has disappeared. It is simplest to characterize this situation by replacing v by 0.

For fission products, a standard new velocity v_0 is assigned to the neutron, and from then on the calculation carried out to determine the new direction in the case of scattering is repeated as often as required.

> Fission: In this case replace v by v_0. According to whether the case in question is that one corresponding to the production of 2, 3, or 4 neutrons, repeat this 2, 3, or 4 times, respectively. This means that in addition to the ρ', s' discussed above, the further ρ'', s''; ρ''', s'''; ρ'''', s'''' may be required.

In the First run, a single cross-section was used to represent fission, and the number of daughter neutrons was determined at random.

> If, however, in the event of collision, fission occurred, a game of chance was played to determine the number of neutrons produced by this collision. The appropriate probability for one, two or three-way fission in the material under consideration was set on the Numerical Function Table. This compared to a random number equidistributed between 0 and 1 made the decision to choose the value of q^* as the appropriate weight which

> this neutron card, when read again, will represent. (In the First Run, p_q^b represented the one, two or three-way fission-probability …) At this point, in the First Run g the generation index, was increased by one. (AR: 20)

In the First run, absorption was represented by a separate cross-section.

> Since in the First Run, Absorption was treated as a separate collision type which happened in the case when the "overdraft" occurred on g_4 following then the positive branch of this sign discrimination, α^* was put equal to 3, thereby noting that Absorption was the terminal event for the neutron under consideration. Both Fission and Absorption branches then joined the point preparatory to printing and from here on followed the common path of the other, (census, escape), terminal events. (AR: 21)

In the second run, however, fission cross-sections were not explicitly recorded, and any neutron that avoided scattering was deemed to engage in a fission event.

> in the Second Run q^* was chosen as ν_{Int} or as $\nu_{Int} + 1$, according to the decision made by comparing ν_{dec} with the random number. In the case of the Second Run q^* could also be 0, since here the fission probability was reduced or, as in some materials, disappeared, due to having "pre-mixed" by hand calculation the fission and absorption cross-sections.

In other words, absorption in the Second run was treated as a special case of fission where the number of product neutrons was 0.

5.1.14 M Print card and restart main loop

First run: boxes 37.1*, 47*, 48*, 49*, 50*.
Second run: boxes 58, 58*, 58°, 58', 59, 60, 61, 62, 63.

In the Richtmyer plan, the computation described ends when a card is printed; in the First and Second runs, however, the computation returns to the start to process the next neutron.

> The appropriate values are then rounded off to five significant figures and all values not yet in position are put in their proper printing locations, i.e. the eight accumulators which are wired to transfer the numerical data from the ENIAC by printing them on IBM cards. At this point, after printing a card $\overline{n}$, the serial number for each printed card is increased by one and the q of the last read neutron card, i.e. the card which path has just been followed, as described above, is re-examined. Since this card could have been either census, fission, or, in the case of the Second run, zonal escape card, the value of q can be any integer $\geq 1 \leq 4$. (AR: 14)

Where the next neutron's characteristics were obtained from depended on the value of q:

> If $q = 1$, then the next card in the reader stack will automatically be read, starting the computation of the path of a new particle. This is achieved with the unconditional transfer I instructing the machine to recommence at the beginning of the logical sequence … If, on the other hand, $q \geq 2$, the path of each particle produced by fission will be separately followed, recommencing the computation with an unconditional transfer K at the point where a new random v and a are assigned to the fission neutron at the given r, i.e. the place where the fission, which produced this neutron, occurred. Before restarting the computation q is reduced by one, thus after having computed the path of the last fission neutron, q becomes 1 and a new card can be read from the stack. Since in the Second run census cards and, even in some cases, zonal escape cards had $q = 2$, the unconditional transfer K was directed to an earlier point at the beginning of the computation, i.e. to the place where fission cards were separated from census and zonal escape cards, as in the latter cases both branches of the "parent" neutron were started with the identical initial data. (AR: 14–15)

This procedure for starting the next iteration was made more complex in the Second run by the requirement to handle the under-critical and super-critical cases differently.

> Since in the Second run a different routine was used at the initial steps of the "virgin" neutron cards of the under-critical assemblies, therefore, until computation on all of these had been finished, the unconditional transfer A was manually substituted to be used for the same purpose as I. (AR: 14)

5.1.15 N Zonal escape

Second run: boxes 79, 80, 81, 82, 83, 84, 85.

In the First run, in the event of zonal escape, the computation continued to follow the course of the neutron into its new zone.

> In the Second Run a card was printed in the event of a zonal escape indicating with the help of two additional symbols for α which type of zonal escape has occurred. In the Second Run the use of a somewhat more detailed treatment of zonal escape was found necessary for the following considerations: Some of the problems in this run were computations on spherical assemblies consisting of 2 or 3 material zones, the outer zone (tamper) having disproportionally large dimensions from the point of view of calculation, i.e. the statistical information obtained through these calculations might have been rendered unwieldy by the size of the outer region. Also because of the very large scattering cross-sections in the tamper, relatively little information regarding other types of collisions could have been obtained between census-cycles in this region. Therefore, for this type of problems, zones of the same material compositions were divided into several zones of varying width and a weighting system was devised in the following manner: Two additional symbols for α^* were introduced to indicate the nature of the zonal escape, if after examining ε it was determined that the neutron particle was travelling "in" towards the center, its weight was doubled, ($q^* = 2$), and $\alpha = 7$ noted the event of zonal escape. If, on the other hand, the neutron was found to be travelling "out" towards the surface of the assembly, a game of chance was played to determine whether or not after the present event the path of the neutron under consideration should be followed further. If the game was "won", α^* again became 7, but $q^* = 1$ – if the game was lost $\alpha^* = 2$, indicating a neutron whose genealogy was ended with this zonal escape and $q^* = 0$ for the same reasons as mentioned in the case of total escape. (AR: 16−17)

The "game of chance", played in box 84, examined whether a particular digit of the most recently computed random number was greater than or less than 5.

> In all cases of zonal escape, we also examined whether the value of i_m, the zone most distant from the centre in which the particle has ever travelled, will have to be substituted by a new value or remain unchanged. After this the path was merged with those of census and Total escape and a card was printed for each neutron having zonal escape as terminal events. (AR: 17)