International Journal of Advanced Nuclear Reactor Design and Technology 5 (2023) 9–16

Contents lists available at ScienceDirect

International Journal of Advanced Nuclear Reactor Design and Technology

journal homepage: www.sciencedirect.com/journal/international-journal-of-advanced-nuclear-reactor-design-and-technology

A criticality assessment through user's routines in FLUKA

F. Vanoni *, E. Padovani, A. Porta, F. Campi, R. Chebac

Politecnico di Milano, Italy

A B S T R A C T

In the present paper, a method to obtain the k_{eff} value using the FLUKA code is described. It took advantage of the chance to write user defined routines in the standard code. A new algorithm was implemented and tested on the simulation of a research nuclear reactor. Results pertaining to the estimation of k_{eff} and neutron fluence are in good agreement with the MCNP's ones.

1. Introduction

The present work takes inspiration from the KCODE algorithm implemented in MCNP [1]. Its purpose is the evaluation of the *criticality condition* of a given multiplying system, like nuclear fission reactors, through the estimate of its *multiplication factor*, also known as *k effective* (k_{eff}). Such objective can be achieved through a controlled development of the neutron population throughout the geometrical domain, obtained by a proper simulation of the *chain reaction* which occurs inside the multiplying region of the system. To understand this concept, one can consider what happens inside a thermal nuclear fission reactor.

- Neutrons are born from fission events inside the fuel region.
- They start travelling and move outside the core fuel. Here, they collide repeatedly with the moderating material until they reach thermal energy (*thermalization*). During this process, a fraction gets lost due to leakage or parasitic absorption.
- Once thermalized, some neutrons manage to return back inside the fuel region, where fission reactions occur and new neutrons are produced. These will undergo the same process of their parents and so on.

If one names the newly born fission neutrons as the new generation and all their parents, together with the lost fraction, the previous generation, the multiplication factor can be defined as follows:

$$k_{eff} = \frac{number\ of\ neutrons\ in\ the\ new\ generation}{number\ of\ neutrons\ in\ the\ previous\ generation} \quad (1)$$

The value of k_{eff} determines the asymptotic behaviour of the system, i.e. exponential decrease, steady state or exponential growth when k_{eff} is < 1, $= 1$ or > 1, respectively.

From the perspective of Monte Carlo codes, the chain reaction poses various problems.

In the field of particle physics, these kind of simulations are employed to study a given system from a statistical perspective; the idea is to generate a large amount of particles which will behave randomly, but always following the physical laws imposed to them. In this way, it is possible to evaluate physical quantities, such as the multiplication factor or the fluence across a surface, by considering the statistical impact of each particle run. So, when one considers a system in which a chain reaction can develop, every generated particle can either get lost or give birth to some secondaries of the same kind, which in turn will undergo the same process, thus leading to unpredictable number of events. In such situation, it is extremely hard to estimate any physical quantity and the computational time becomes huge even for a small number of source particles.

In the following, the KCODE solution to the problem is briefly presented. It can be subdivided into few passages.

- When the first cycle of the simulation starts, neutrons are born in a generic location in the geometrical domain specified by the user. These neutrons belong to the first generation. When they produce fission reactions, data about secondary neutrons (in particular kinetic energy and birthplace coordinates) are collected and their transport is stopped. All these secondaries correspond to the second generation. A first evaluation of the multiplication factor is thus simply the ratio between the number of neutrons in the second generation and the number of those in the first one.
- The second cycle employs the data of the second generation to initialize source neutrons, so fission reactions will give birth to the third generation. The process is iterated a certain number of times; k_{eff} is evaluated once per cycle by applying the formula in (1),

* Corresponding author.
E-mail address: fabio.vanoni@polimi.it (F. Vanoni).

https://doi.org/10.1016/j.jandt.2023.02.001
Received 12 December 2022; Received in revised form 26 January 2023; Accepted 20 February 2023
Available online 26 February 2023

corrected for a factor introduced to eliminate trends in the number of fission neutrons. The estimate becomes more and more accurate as cycles go by since the birthplace and energy distribution of neutrons becomes every time more coherent with the physical system.

The algorithm, together with a rich cross-section library for every element, provides a very accurate estimate of k_{eff}. Its calculation is performed also by computing the ratio of other neutron-related quantities between successive generations, such as collisions, absortions and track lengths; the formula is the same reported in (1), but considering, respectively, the total amount of collisions undergone by all neutrons, the total number of neutron absorption and the total length of all the tracks covered by neutrons during their transport. These estimates provide overall better results, given the more accurate statistic these quantities carry.

The present work aims at realizing an algorithm which, as KCODE, can reproduce the physical behaviour of multiplying systems inside the FLUKA environment [2,3], by taking advantage of the numerous user routines supplied by the code. This work does not aim at replacing MCNP or other well-known codes in the criticality calculation field, but instead wants to cover a purely didactic scope, by also taking advantage of its open source nature.

Before proceeding, a brief presentation of the FLUKA code is reported.

FLUKA is a multi-purpose particle transport code; its applications are many, including high energy physics, shield design, dosimetry and so on. For this reason, it is able to reproduce a vast number of phenomena in the field of particle physics. The other strength of FLUKA is its versatility in allowing users to implement their own code by employing a vast set of *sub-routines*. Those are called by the main routine during standard operations, each at a specific moment of the run; some are executed automatically and some only if a certain option (usually specified in the input file) is activated. Their purpose normally depends on when they are executed; for example, a routine called just after the definition of the source particle and before its transport begins may be used to override some of its characteristics with others which are too case-specific to be among the standard options.

2. Algorithm description

An exhaustive description of the newly developed algorithm, called *FLUKeff*, is now presented. However, before proceeding, it is worth mentioning the two methods that the code adopts to transport what in FLUKA are referred to as low energy neutrons, i.e. with kinetic energy below 20 MeV.

- *Group-wise* transport treats neutron moderation as a discrete process by dividing the neutron energy range into discrete intervals. In FLUKA, the default option is 260 group of approximately equal logarithmic width between 20 MeV and 10^{-5} eV. Among these groups, 31 are thermal. During neutron moderation, elastic and inelastic reactions are simulated by group-to-group transfer probabilities: the result is a jump from a kinetic energy representing one group to another value corresponding to another group in a completely discrete way. This approach is fast and reliable in most situations, but does not preserve correlations between products and may be unfit when the fine structure of resonances is relevant. Energy and momentum are conserved on average.
- *Point-wise* transport, instead, treats neutron moderation as a continuous process, thus ensuring an accurate reproduction of resonances. Moreover, correlation among products is preserved and kinetic energy and momentum are conserved exactly at each neutron interaction. The drawback is a much longer computational time.

Another difference between the two can be noticed when considering how the code treats fission neutrons. When using the former, they are all classified as secondary particles and as such they are stored in the same stack. On the other hand, when using point-wise transport, one of the fission neutrons is treated as a new primary particle, therefore it can't be found stacked together with its "brothers", as it is immediately transported. Such fact had a sensible impact on the development of the algorithm since one of the user-routines employed is dedicated only to the solution of this discrepancy.

2.1. Presentation of the employed sub-routines

Before describing the algorithm structure, a brief presentation of the five FLUKA routines involved is reported, together with their scope.

2.1.1. mdstck.f

This routine is automatically called after each nuclear interaction in which at least one secondary particle has been produced and before they are loaded into the main stack for transport.

It covers three purposes in the algorithm.

- When fission occurs, it stops the transport of all the secondaries born from the interaction by setting their statistical weight to zero; in this way, the occurrence of an undesired chain of reactions is prevented.
- It flags the occurrence of fission events for the sake of routine *usrmed.f*; the reason will be cleared later on.
- It keeps the count of every neutron born from fission; moreover, it writes their energy and the spatial coordinates of the reaction to a text file called *new_gen* in order to use such data to generate source neutrons in the following cycle.

2.1.2. source.f

This routine is activated by inserting the card SOURCE in the input file. It is called at the beginning of each event, just before the transport of the source particle. It allows to implement or modify any characteristic of the latter which is too case-specific to be declared by cards BEAM, BEAMPOS or POLARIZA.

In the present work, *source.f* allows the use of data from fission neutrons to generate source particles by reading energy and coordinates from a text file called *old_gen*, which contains data of neutrons born during the previous cycle; in the case of the first cycle, when the latters aren't yet available, the user can decide a first-try energy by employing the first WHAT of card SOURCE and a first-try position by means of card BEAMPOS. In both cases, the particle direction is sampled isotropically.

2.1.3. usrini.f

The routine is activated by card USRICALL and is called at the beginning of each cycle of the current run.

Its purpose is to flag the existence of file *old_gen* for *source.f* in order to discriminate whether the current cycle is the first one or not. Only when such case occurs, the routine also creates the *new_gen* file for *mdstck.f* and *usrmed.f*.

2.1.4. usrmed.f

This routine is activated by card MAT-PROP with SDUM = USERDIRE. The card also allows the user to flag some materials; the routine is then called every time a particle is going to be transported in one of those flagged materials or from one to another.

As previously mentioned, there is a discrepancy in the treatment of secondary neutrons when considering group or point-wise transport; *usrmed.f* is employed to overcome such problem. In the algorithm, it is activated only when a fission reaction occurs to stop the single neutron which escapes from the control of *mdstck.f* when using the point-wise treatment; it proceeds then to write its data to *new_gen* and update the fission neutron counter.

2.1.5. usrout.f

The routine is activated by card USROCALL and is called at the end of

each cycle of the current run.It covers three purposes in the algorithm.

- It calculates an estimation of k_{eff} by dividing the total number of fission neutrons born during the cycle by the number of source particles run per cycle; the number is the same specified in the first WHAT of the START card and is passed to usrout.f by reporting the same value in the first WHAT of card USROCALL.
- It writes the above-mentioned estimation to a text file called *keff*, together with the average value of k_{eff} over all cycles and the relative standard deviation. By means of the second WHAT of card USROCALL, the user can also specify a number indicating from which cycle the calculation of the average value will start; such feature allows to discard the first cycles, meant only to allow the neutron population to reach the asymptotic profile.
- It deletes the *old_gen* file since its content is no longer needed, then proceeds to replace it with the *new_gen* file by changing its name, so that to advance neutron generations between cycles.

2.1.6. NEUDAT

Apart from the just introduced routines, a new common, called NEUDAT, is required to complete the algorithm. Its purpose is to store variables which will be used by more than one routine during each cycle, as for example the various flags and the fission neutron counter.

To summarize, a brief description of all sub-routines employed is proposed in Table 1.

2.2. Algorithm structure

After introducing all the routines, the algorithm working principle is reported so to highlight how they interact.

1. At the beginning, routine *usrini.f* is called to check for the existence of the *old_gen* file; the operation is thus repeated once per cycle.
2. If *old_gen* exists, *source.f* will read the data to initialize source particles directly from it; in the opposite case, it will consider the entries from cards in the input file, as specified in 2.1.2. If the number of particles to run per cycle exceeds the number of entries in *old_gen*, the file is simply rewound; since every particle starts moving in a random direction, events will be unique in any case.
3. When fission reactions occur, *mdstck.f* (and also *usrmed.f* when using point-wise transport) will stop every secondary, store fission neutron data and update the neutron counter. Basically, the occurrence of a fission reaction marks the end of the current event in order to prevent chains of undesired interactions.
4. When all particles have been run, *usrout.f* closes the cycle by calculating k_{eff}, its average value and the associated standard deviation, then writes everything to the *keff* file. Finally, it deletes *old_gen* if it exists and renames *new_gen* accordingly, as specified in 2.1.5.

In conclusion, a flux diagram showing the structure of one cycle is reported in Fig. 1.

Table 1
Summary of the sub-routines employed and their main purpose.

Sub-routine	Main purpose
mdstck.f	To stop secondary neutrons and store their data for the next cycle
source.f	To generate source neutrons either by reading from a file or from the input
usrini.f	To check the existence of *old_gen* and create *new_gen* when needed
usrmed.f	Same as *mdstck.f*, but only when using point-wise transport
usrout.f	To calculate k_{eff} and other related quantities and setup files for next cycle

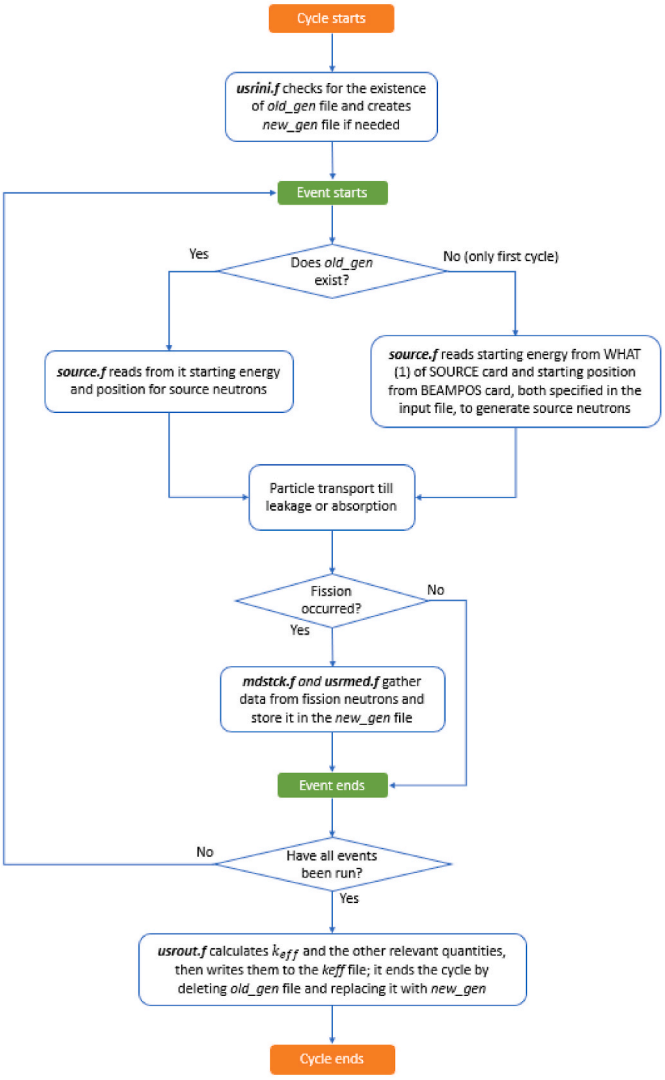


Fig. 1. Flux diagram showing the structure of one cycle.

3. Simulations

FLUKeff in FLUKA has been tested and compared to MCNP's KCODE by considering as a model the Politecnico di Milano's research nuclear reactor, L-54 M. Every information regarding geometry and materials composing the structure has been recovered from Refs. [4–9].

3.1. L-54 M research nuclear reactor

L-54 M is a homogeneous-fuel thermal research nuclear reactor fueled by an uranyl sulfate aqueous solution with an enrichment level of 19.94%. Its core consists of a sphere of AISI 347 nuclear grade stainless steel, 20 cm in diameter and 1.85 mm in thickness. It is penetrated by five stainless steel tubes: one horizontal exposure tube (the "glory hole"), and four vertical channels for control rods. On top of the sphere, an overflow chamber is soldered to contain the liquid fuel in case of excessive power excursions. the latter is also composed of the same stainless steel. Fig. 2 shows the reactor core structure.

Fig. 2 reports also the interior of the reactor core, exposing the cooling system of the structure. The latter is composed of a series of complex stainless steel coils in which water flows to remove the heat. Given the difficulties in simulating such a complex geometry, it was decided to treat the interior of the core as a homogeneous mixture of liquid fuel, stainless steel and water. Another complex element which

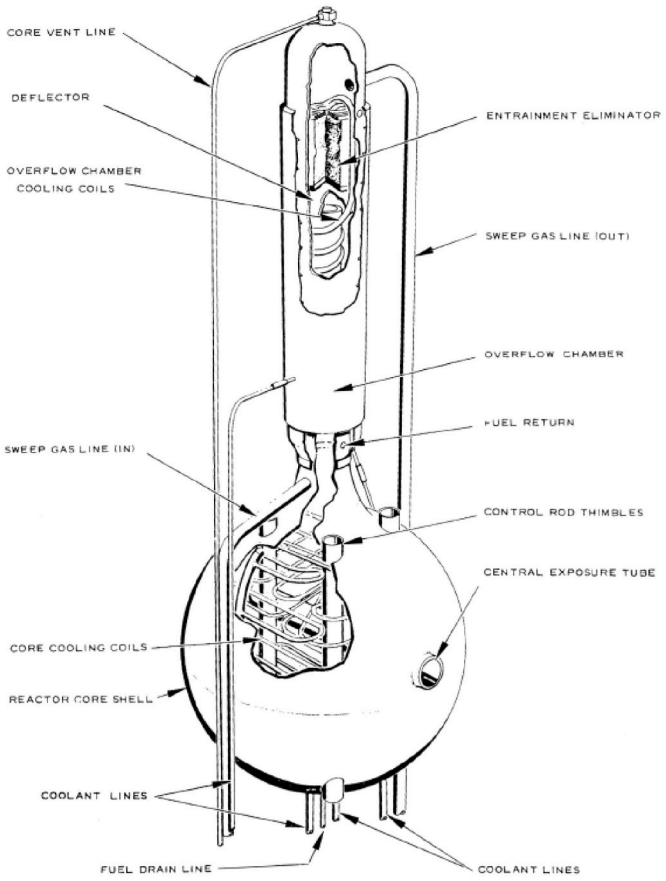


Fig. 2. Picture of L-54 M's core.

was simplified is the overflow chamber: it was treated as a simple stainless steel cylinder. Indeed, as simulations will show, such approximations won't affect the goodness of results.

Both the moderator and the reflector of the reactor are made of AGOT grade graphite. The former is disposed around the core, inside a cylindrical aluminium case; its shape was thought to best fit between the two components. The latter is outside the aluminium case and is shaped as a parallelepiped; as for the moderator, also the reflector is covered in an aluminium box. Lastly, a heavy concrete shield encloses the entire structure.

Fig. 3a reports a horizontal section of the entire reactor; it is also possible to distinguish the various exposure tubes employed for irradiation purposes. Fig. 3b, instead, is the representation of L-54 M realized with *Flair* [10], a graphical user interface for FLUKA. In particular, it is the *xy* plane crossing the center of the core.

By looking at Fig. 3, two major differences can be noticed: the profile of the concrete shield has been simplified and only some of the exposure channels have been considered. As for the core and the overflow chamber, these simplifications will not jeopardize the goodness of results.

3.2. Premise on cross-sections

In FLUKA, point-wise, group-wise and an hybrid case were tested. Point-wise cross-sections which consider molecular binding of hydrogen, deuterium and carbon into, respectively, light water, heavy water and graphite were also employed. However, it is important to mention that the FLUKA manual discourages the use of such cross-sections when neutrons descend under the threshold of 3 eV, i.e. when they enter the *thermal region* of the energy spectrum. Indeed, it suggests to resort to the group-wise treatment in such scenario. Nevertheless, for the sake of completeness, these cross-sections were tested also in the complete spectrum, concurrently with full point-wise transport. The last one has been recently introduced with FLUKA21.2, which is the first update of the code providing a point-wise cross-section library complete of all the elements and their most common isotopes.

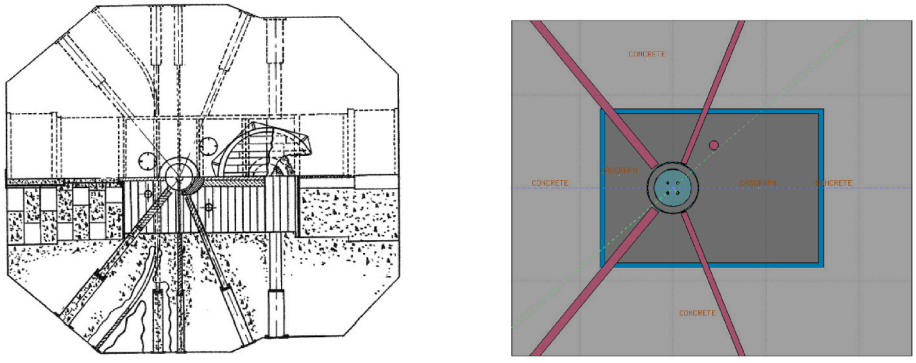
Finally, it must be clarified that cross-sections in MCNP have been chosen with the scope of maximizing the accuracy in estimating k_{eff}, discarding the idea of using the exact same libraries in the two codes. Such decision is in favor of a comparison in which both algorithms are employed at their best.

3.3. Multiplication factor

A comparison on the estimate of the multiplication factor, called k_{eff}, is now presented. Three reactor configurations will be considered: critical, subcritical and supercritical.

Before proceeding, two limitations of FLUKeff must be highlighted. The first one is strictly connected to the algorithm: the estimate of k_{eff} gives reasonable results only when performing *fully analogue* simulations, i.e. every source particle is born with weight equal to 1, which remains constant during the transport. Moreover, any secondary particle born from the latter will also have the same weight. The second limitation pertains to the FLUKA code itself: when fission occurs, both codes will consider delayed and prompt neutrons as if they were all prompt. While MCNP has the opportunity to properly generate delayed neutrons, FLUKA currently lacks such capability. Anyway, this feature wouldn't be of any help for the current study.

Results are reported in Table 3, where Pw and Gw stand, respectively, for point-wise and group-wise cross-sections and bth indicates the thermal region when considering molecular binding for H, D and C (see Subsection 3.2). Under the column reporting variations from the main case, symbols must be interpreted as follows: before the colon, the



(a) Real (b) Flair

Fig. 3. Real section of L-54 M compared with its representation in Flair.

Table 2
k_{eff} obtained with KCODE by employing Point-wise cross sections and considering different reactor configurations.

Case	Configuration	Particles/cycle	Cycles	k_{eff}	St. dev.
Pw	cr	50000	100	0.99786	0.00035
Pw	subcr	50000	100	0.96750	0.00040
Pw	supcr	50000	100	1.01574	0.00041

alternative cross section is reported; after the colon, the materials involved are specified. For example, Gw:bth means that the group-wise treatment is reserved only to the thermal region for the above-mentioned bound materials. Each simulation has been run neglecting the first 30 cycles from the calculation of the average k_{eff} value; to determine the correct number of cycles to skip, a trial and error approach is required by considering how the k_{eff} value varies cyclewise.

The KCODE counterpart returns the following k_{eff} values for the three configurations (Table 2). Also in this case, the first 30 cycles have been neglected. The point-wise cross sections used are taken from ENDF/B-VII at room temperature.

By looking at Table 3, a series of considerations must be pointed out.

- The best result is obtained by considering full point-wise transport, thus employing point-wise cross-sections contemplating molecular binding also in the thermal region.
- Given the same total number of particles, standard deviation seems to benefit more from shorter, but more numerous, cycles; it can be noticed by comparing results of 250000x100 and 50000x500 simulations.
- Deltas in the value of k_{eff} given by different configurations are coherent with the ones obtained with KCODE.

For the sake of a direct comparison in terms of computational time and efficiency, the same machine has been employed to run all of the above-mentioned simulations. The computer was provided with twelve Intel® Core™ i7-3930K CPU @ 3.20 GHz processors and 64 GB of RAM, even though parallelization was not employed; in its current state, FLUKeff does not support such feature. Point-wise transport was adopted. To correctly assess which code is the most efficient, the *Figure of Merit* defined by expression (2) must be considered.

$$FoM = \frac{1}{r^2 t} \tag{2}$$

where r is the relative error, i.e. the ratio between standard deviation of the mean and the mean itself, and t is the computational time. Table 4 resumes all data.

In conclusion, it is evident that the prediction of MCNP is for sure the most accurate and efficient. Such result must not surprise though, since KCODE is an algorithm which was born with this scope and has been refined over the years. Moreover, the cross section library of MCNP is much richer than the one of FLUKA, even after the recent addition of the point-wise library. However, results obtained with FLUKeff are undoubtedly satisfying, in particular considering the didactic purpose of this work; the maximum difference between the two algorithms is just around 3% and goes down to 1.4% in the best case scenario.

3.4. Neutron fluence

A comparison between the neutron fluence, both thermal and total, calculated by the two algorithms has been carried out. Differently from k_{eff}, the estimator of this quantity is already implemented in both codes: these are the card FMESH4 in MCNP and USRBIN in FLUKA. Both are used to score different quantities over a mesh-like, three-dimensional spatial structure, called mesh tally or binning detector, superimposed to a portion of the geometrical domain. In the present case, the region of the reactor under examination was the graphite reflector and its interior, which includes the graphite moderator, the core and the overflow chamber.

Regarding the scoring in FLUKA, one more step is required to get accurate results. By means of routine *fluscw.f*, it is possible to activate a weighting function which will be applied to the output of the scoring; in this case, such weight is necessary to discriminate the contribution of thermal neutrons from the total fluence, obtaining the above-mentioned thermal fluence. Routine *fluscw.f* is activated by the USERWEIG card, particularly by assigning a value greater than 0 to the third WHAT. Also for the scoring, it is necessary to neglect the first cycles, particularly the same number discarded for the k_{eff} estimation. The task is fulfilled by a simple MATLAB [11] script.

It must be mentioned that both codes calculate neutron fluence by considering the so-called *track-length*, i.e. the sum of the length of all

Table 3
k_{eff} obtained with FLUKeff by employing Point or Group-wise cross sections and considering different reactor configurations.

Main case	Variations	Configuration	Particles/cycle	Cycles	k_{eff}	St. dev.
Pw	Gw:bth	cr	50000	100	1.02224	0.00079
Pw	Gw:bth	cr	50000	500	1.02101	0.00031
Pw	Gw:bth	cr	250000	100	1.02099	0.00035
Pw	–	cr	50000	100	1.01389	0.00091
Pw	–	cr	50000	500	1.01397	0.00030
Pw	–	cr	250000	100	1.01477	0.00032
Gw	–	cr	50000	100	0.97042	0.00094
Gw	–	cr	50000	500	0.96946	0.00032
Gw	–	cr	250000	100	0.96972	0.00033
Pw	Gw:bth	subcr	50000	500	0.98997	0.00032
Pw	–	subcr	50000	500	0.98161	0.00031
Gw	–	subcr	50000	500	0.94058	0.00031
Pw	Gw:bth	supcr	50000	500	1.03769	0.00030
Pw	–	supcr	50000	500	1.03222	0.00031
Gw	–	supcr	50000	500	0.98588	0.00033

Table 4
Time and efficiency comparison between KCODE and FLUKeff.

Code	Case	Configuration	Particles/cycle	Cycles	k_{eff}	St. dev.	Time (min)	FoM ($\cdot 10^3$)
KCODE	Pw	cr	50000	100	0.99786	0.00035	97	83.80
FLUKeff	Pw	cr	50000	100	1.01389	0.00091	150	8.28

tracks covered by neutrons in the unit volume. Its unit of measure is thus cm/cm^3, which is coherent with the standard fluence dimension, namely 1/cm^2. Moreover, both codes tend to normalize scored quantities by the number of source particle run, in favor of a clear comparison between simulations of different length or from different codes. For such reason, the correct unit of measure is then cm/cm^3/*sp*, where *sp* stands for source particle. Finally, for the sake of a simpler interpretation of results, the mesh in both codes is composed of cubic cells of 1 cm^3 each.

The post-process is executed in MATLAB, where function imagesc is employed to create color-scaled plots of 2D slices of the 3D matrix resulting from scoring. In particular, the *xy* plane crossing the center of the core (the same reported in Fig. 3b) has been chosen to show an overview of the fluence distribution inside the reactor. Fig. 4 reports the results of the scoring of both thermal and total fluence for all three configurations. Logarithmic scale has been employed since fluence values span across many orders of magnitude.

Fig. 4 shows also the difference in the position of the four control rods between the three configurations, which can be particularly noticed while looking at the thermal fluence.

In order to have a quantitative comparison between the two codes, the relative difference between results from KCODE and FLUKeff has been calculated. The formula reported in (3) has been considered for the calculation.

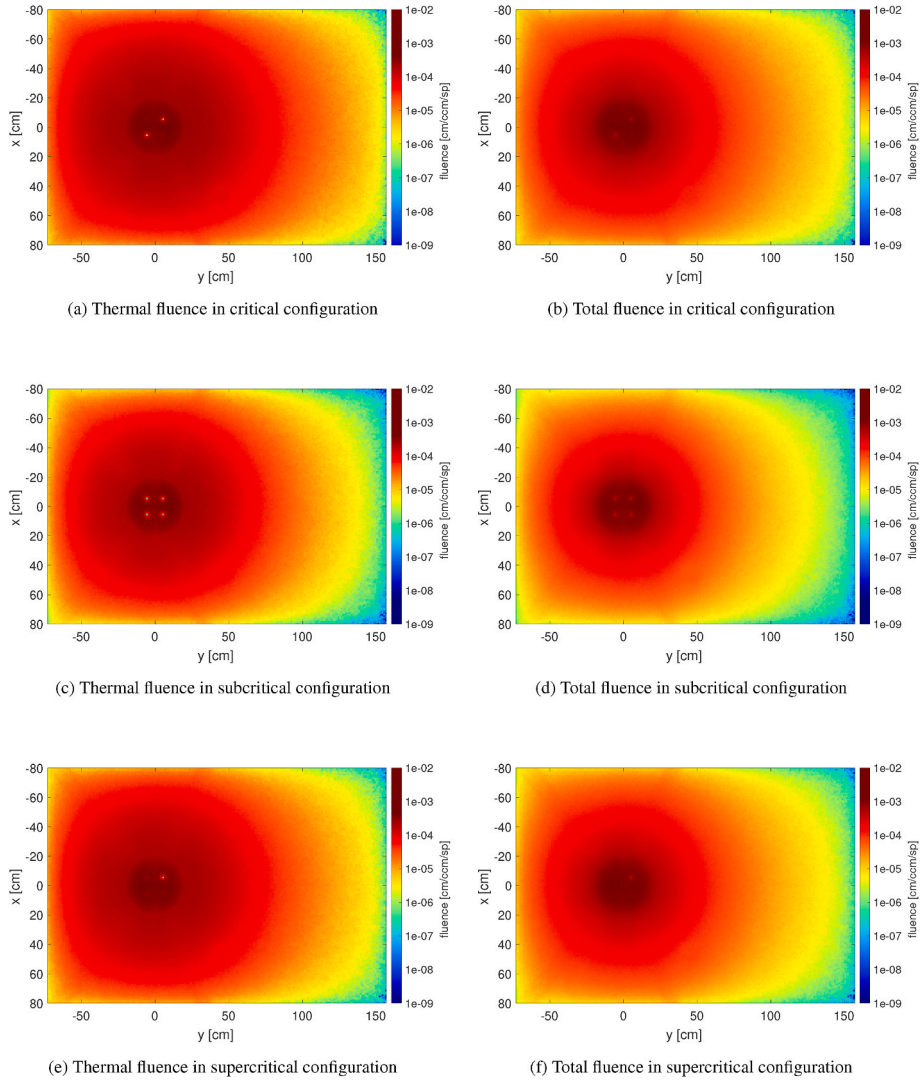
$$diff(x_i, y_j, z_0) = \frac{\Phi_{FLUKA}(x_i, y_j, z_0) - \Phi_{MCNP}(x_i, y_j, z_0)}{max(\Phi_{FLUKA}(x_i, y_j, z_0), \Phi_{MCNP}(x_i, y_j, z_0))} \tag{3}$$

Φ indicates the neutron fluence, while subscripts i and j scan the entire slice. It can be noticed that the relative difference, as formulated in (3), can assume both positive or negative values. In particular, when it is negative, the fluence estimated by FLUKeff is lower than the one estimated by KCODE.

To show the comparison in the clearest way possible, the idea of a color-mapped slice has been discarded in favor of a more readable plot; for this purpose, a segment has been selected inside the geometry, particularly the one starting from the origin, which corresponds to the core center, and ending 1 m away in the positive y direction. All the values of the relative difference along its path have been plotted for all the six cases and reported in Fig. 5. Here, the difference among the three cross-section approaches proposed in Table 3 is also considered.

Regarding the total fluence, Fig. 5 shows that the difference hardly exceeds the ±10% interval, independently from the cross-sections employed; some major discrepancies arise only when moving further from the core, where they are bound to occur and are thus of less interest.

Regarding thermal fluence, instead, one can notice that the best result comes from using point-wise cross-sections, with the exception of the thermal region for bound materials. On the other hand, the point-



(a) Thermal fluence in critical configuration

(b) Total fluence in critical configuration

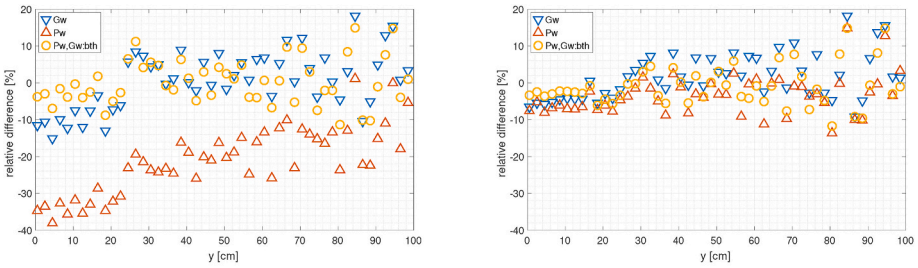
(c) Thermal fluence in subcritical configuration

(d) Total fluence in subcritical configuration

(e) Thermal fluence in supercritical configuration

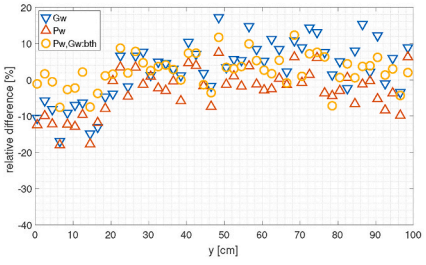
(f) Total fluence in supercritical configuration

Fig. 4. Color maps of thermal and total fluence obtained in FLUKA by using FLUKeff.

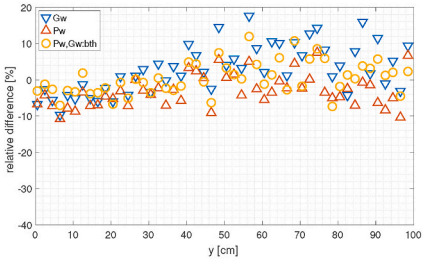


(a) Thermal fluence in critical configuration

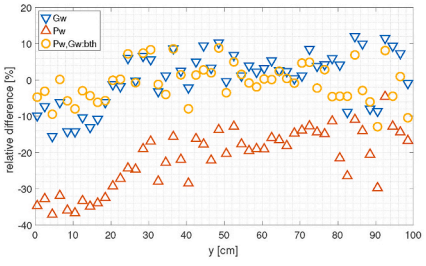
(b) Total fluence in critical configuration



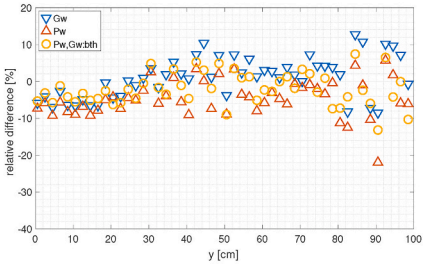
(c) Thermal fluence in subcritical configuration



(d) Total fluence in subcritical configuration



(e) Thermal fluence in supercritical configuration



(f) Total fluence in supercritical configuration

Fig. 5. Plots of the relative difference between thermal and total fluence obtained from KCODE and FLUKeff.

wise treatment is arguably the worst, especially inside the 20 cm range, which represents the core radius.

One reason for the discrepancy of results within the core region might reside in the water cross-sections, which may present differences when switching from one library to another. To confirm or discard such hypothesis, a simple simulation of a water sphere at room temperature with an isotropic point source of mono-energetic thermal neutrons (0.025 eV) at its center has been performed in both codes. The same cross-sections considered in the present work have been employed. The comparisons of the neutron fluence is reported in Fig. 6.

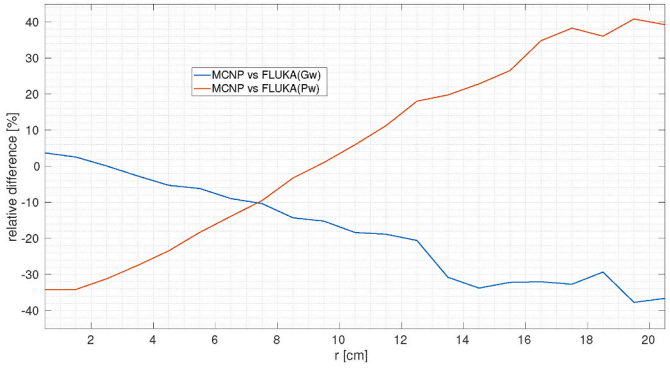


Fig. 6. Comparison between MCNP and FLUKA fluences for different water cross-sections.

In FLUKA, both Gw and Pw cases were tested. The former gave reasonable results, starting from approximately a 4% difference at the center. The latter, instead, shows an excessive offset at the start, thus confirming the suspects on the water cross-sections; moreover, the trend is very similar to the one in Fig. 5a and (c). Such fact confirms also what has been mentioned in Subsection 3.2 regarding the use of point-wise cross-sections contemplating molecular binding in the case of thermal neutrons. Indeed, it is advisable to avoid their use and resort to the group-wise treatment in such scenario.

3.5. Shannon entropy convergence

To confirm the goodness of the algorithm, the calculation of the Shannon entropy of the neutron source has been attempted.

This quantity, usually called H, represents the amount of information, or uncertainty, inherent to all possible outcomes of a random variable. It can be calculated as follows:

$$H(X) = -\sum_{x} p(x)\ log_b(p(x)) \tag{4}$$

where X is a random variable, the collection of all the x are its possible outcomes and $p(x)$ their related probabilities, so that $\sum_x p(x) = 1$; b is the base of the logarithm, which is usually chosen among 2, e and 10; base 2 gives the unit of *bits*, while base e gives units of *nats* and base 10 units of *dits*. For the sake of clarity, the case of the coin toss can be considered as an example; its related Shannon entropy, calculated by using base 2 logarithm, is reported in (5).

$$H(coin\ toss) = -p(heads)\ log_2(p(heads)) - p(tails)\ log_2(p(tails)) = -0.5\ log_2(0.5) - 0.5\ log_2(0.5) = 1 \quad (5)$$

As a comparison, in (6) the case of a biased coin (70% heads and 30% tails) is reported.

$$H(biased\ coin\ toss) = -0.7\ log_2(0.7) - 0.3\ log_2(0.3) \approx 0.8816 \quad (6)$$

By comparing the two results, it can be immediately noticed that the Shannon entropy in the second case is lower. The reason behind such result is that the outcome of a biased coin toss is more predictable, and thus less uncertain. One can say that it is a "more ordered" random variable than a fair coin toss. To further clarify this concept, in the case of a totally biased coin (100% heads and 0% tails), the entropy is $log_2(1) = 0$ since there is no uncertainty at all.

The Shannon entropy of the neutron source is automatically calculated by KCODE to establish which is the minimum number of cycles to be discarded from the k_{eff} estimation. The idea is to evaluate the "order" of the spatial distribution of source points, cycle by cycle; indeed, the entropy convergence to a fixed value indicates that such distribution is not changing the amount of information it carries, i. e., it has reached its asymptotic shape. When it happens, the scoring of physical quantities can begin.

To calculate the entropy, KCODE refers to the position of source points inside a 3D grid. The user can define the dimension and location in the geometrical domain of such grid in the input file, but it is not mandatory since the algorithm is able to place and shape it properly by simply running the simulation and analyzing the first data produced.

In the case of FLUKeff, instead, a way to calculate the entropy have to be implemented. The realization of a complex algorithm, such as the one of KCODE, is beyond the scope of this work; the aim is thus to ensure the convergence at least for the presented case. The process can be divided in few steps.

1. The first problem is the definition of a random variable containing significant information for a spatial distribution of points. Giving the spherical symmetry of the core region, the distance of source points from the core center has been chosen.
2. Since such variable assumes continuous values, it must be discretized in order to use eq. (4) for the calculation. So, the 20 cm core radius has been divided in 20 intervals, each 1 cm in length. To clarify, a reference to the quantities presented in the above-mentioned formula is reported.
   - X: distance of a source point from the core center.
   - x: source point falling in one out of the 20 intervals defined above.
   - $p(x)$: probability of falling in a specific interval.
3. Finally, the process has to be implemented in the code. The most simple way is to define a 20 elements array, where each element is the counter for neutrons born at a distance from the core center equal to the element's index, once truncated to an integer. The calculation of the distance and the relative counter update are executed the same moment neutron data are stored in the *new_gen* file, by either *mdstck.f* or *usrmed.f*. At the end of the cycle, *usrout.f* calculates the Shannon entropy considering the array of counters normalized to the total number of generated neutrons and prints the value in the *keff* file along the other quantities. Base *e* logarithm has been chosen, but it can be any.

Fig. 7 shows the comparison between k_{eff} and Shannon entropy convergence, calculated during a simulation of 50 cycles, 50000 source particles each, employing group-wise transport and in critical configuration.

By looking at Fig. 7, it can be noticed that there is good coherence between the converge of the two quantities, which happens around the sixth cycle. Such fact proves the goodness of the algorithm in simulating at least this physical system, thus the goodness of all results previously reported.

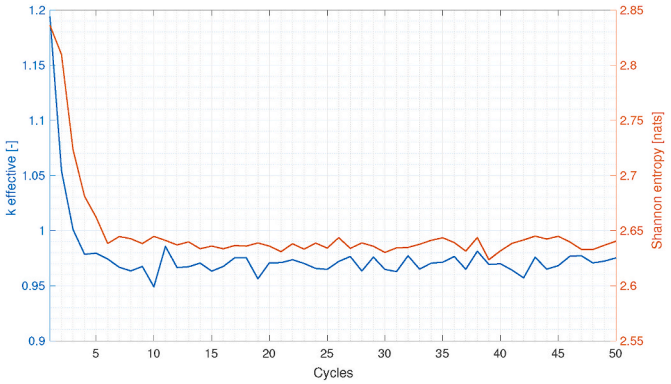


Fig. 7. Comparison between k_{eff} and Shannon entropy convergence.

4. Conclusion

In the present work, FLUKeff has been presented as a new and open source tool to simulate multiplying systems in FLUKA. Its structure has been explained and its capabilities compared with KCODE by MCNP on the simulation of a real experimental nuclear reactor, namely Politecnico di Milano's L-54 M. Major effort was placed on the estimate of k_{eff} and neutron fluence, both thermal and total. Results have been reported and discussed. Moreover, the new and first point-wise cross-sections library of FLUKA has been employed and compared with the already-existent group-wise treatment.

As already mentioned at the end of Subsection 3.3, it is clear that FLUKeff can't be on par with KCODE in the calculation of k_{eff}; indeed, it can't substitute the latter as a reference tool for criticality estimation. Nevertheless, the capability of the new algorithm to reproduce the physical behaviour of a multiplying system is surely satisfying, thus justifying its use as a didactic tool.

References

[1] C.J. Werner, et al., MCNP Users Manual - Code Version 6.2, Los Alamos National Laboratory, 2017 report LA-UR-17-29981.
[2] T.T. Bohlen, F. Cerutti, M.P.W. Chin, A. Fassò, A. Ferrari, P.G. Ortega, A. Mairani, P.R. Sala, G. Smirnov, V. Vlachoudis, The FLUKA code: developments and challenges for high energy and medical applications, Nucl. Data Sheets 120 (2014) 211–214.
[3] A. Ferrari, P.R. Sala, A. Fassò, J. Ranft Fluka, A Multi-Particle Transport Code. CERN-2005-10, 2005. INFN/TC_05/11, SLAC-R-773.
[4] Atomics International, Disegni di progetto del reattore l-54m, 1959.
[5] J.J. Lee, et al., The Density of Uranyl Sulfate Solutions and the Determination of Uranium Concentration by Density Measurements, Technical report, 1952.
[6] W. L. Marshall. Density-weight Percent Molarity Conversion Equations for Uranyl Sulfate - Water Solutions at 25.0 °c and between 100-300 °c. Oak Ridge National Laboratory.
[7] R.E. Nightingale, Nuclear Graphite, first ed., Academic Press, Cambridge, USA, 1962, ISBN 9781483228549.
[8] E. Orban, et al., Physical properties of aqueous uranyl sulfate solutions from 20 to 90°c, J. Phys. Chem. 60 (4) (apr 1956) 413–415.
[9] S. Terrani, Il reattore e gli impianti nucleari del cesnef - rapporto di sicurezza, Centro Studi Nucleari Enrico Fermi, 1961.
[10] V. Vlachoudis, FLAIR: a powerful but user friendly graphical interface for FLUKA, in: Proc. Int. Conf. On Mathematics, Computational Methods & Reactor Physics (M&C 2009), 2009. Saratoga Springs, New York.
[11] Natick, Massachusetts, The Mathworks, Inc., 2021. MATLAB version 9.10.0.1613233 (R2021a).