CBPF-NF-063/01

The MCEF code for Nuclear Evaporation and Fission calculations

A.Deppman[a], O.A.P.Tavares[b], S.B.Duarte[b], E.C. de Oliveira[b], J.D.T. Arruda-Neto[a,c], S.R. de Pina[a], V.P. Likhachev[a], O. Rodriguez[a,d] , J. Mesa[a], and M. Gonçalves[e1].

[a)]Instituto de Física da Universidade de São Paulo

P.O.Box 66318 - CEP 05315-970

São Paulo, Brazil

[b)]Centro Brasileiro de Pesquisas Físicas -CBPF/MCT

22290-180 Rio de Janeiro, Brazil

[c)]Universidade de Santo Amaro - UNISA

São Paulo, Brazil

[d)]Instituto Superior de Ciencias y Tecnología Nucleares - ISCTN

A.P. 6163 La Habana, Cuba

[e)]Instituto de Radioproteção e Dosimetria - IRD/CNEN

22780-160 Rio de Janeiro, Brazil

Abstract

We present an object oriented algorithm, written in the Java programming language, which performs a Monte Carlo calculation of the evaporation-fission process taking place inside an excited nucleus. We show that this nuclear physics problem is very suited for the object oriented programming by constructing two simple objects: one that handles all nuclear properties and another that takes care of the nuclear reaction. The MCEF code was used to calculate important results for nuclear reactions, and here we show examples of possible uses for this code.

PACS numbers: 24.75.+i, 25.85.-w, 21.60.Cs

[1]Also: Sociedade Educacional São Paulo Apóstolo- UniverCidade, 22710-260 Rio de Janeiro, Brazil.

PROGRAM SUMMARY

- Title of program: MCEF
- Catalogue number:
- Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland.
- Licensing provisions: none
- Computer for which the program is designed and others on which it has been tested: Microcomputer with Pentium Intel 233. Installation: Institute of Physics, University of São Paulo, Brazilian Center for Physical Research -CBPF/MCT.
- Operating system: MS-DOS 6.00, Windows 95 and Windows NT 4.0, Linux
- Program language used: Java 1.2.
- Memory required to execute with typical data: 8 Mbytes of RAM memory and 4 Mb of Hard disk memory (.class files only).
- No. of bits in a word: 16
- No. of lines in distributed program, including test data, etc.:363.
- Keywords: Photonuclear Reactions, Evaporation, Fission, Monte Carlo Method.
- Nature of the physical problem: The competition between evaporation and fission is an important problem in intermediate and high energy nuclear reactions. Here, neutrons, protons, alphas and possibly other particles escape from the excited nucleus in competition with the fission process. Considering the fact that an exact calculation including all the channels in this nuclear reaction could be a complex problem, a statistical description of all these possible decaying modes could be a major simplification. In fact, rough analytic approximations are possible, and the results are in reasonable agreement with the experimental data for relatively low

reaction energies, where the most important reaction channels are just the neutron evaporation and fission. At higher energies, other channels, like proton and alpha particle evaporation, become important, and the analytical calculation of the reaction process is much more complex.

- Method of solution: The most reasonable way to avoid the complexity of these problems, and to obtain the required data is the Monte Carlo (MC) simulation. In fact, this method is particularly suited for processes which are intrinsically statistical in their nature, as the nuclear reactions described above. Also, the object oriented approach for the algorithm is useful for solving this problem, since it is appropriate for the atomic nucleus problems, and turns the MC calculation clear stated problem.

- Typical Running Time: depends on the desirable statistical accuracy. For the particular set of initial parameters shown as an example in this work, the running time is approximately 5 seconds.

LONG WRITE-UP

I Introduction.

Nuclear physics problems are particularly suited for the application of the object oriented (OO) paradigm in the software programming. The OO software is appropriate whenever it is possible to fragment the algorithm into many well defined objects which have some parameters that one can control or access from outside, and which may modify the object's characteristics. In nuclear physics we can find many algorithms with these features, and in particular, the atomic nucleus itself can be easily treated as an ordinary object interacting with its environment through some specific set of parameters that can be modified from outside by changing those parameters.

The OO programming presents many advantages, such as the readability, good maintainability, reusability and extensibility[1] of the program. The Java language, which has been created specifically following the OO paradigm, has the additional feature of portability, allowing the possibility to easily run the same program at different machines[2].

In the Nuclear Physics field, the MC calculation is frequently used since it allows the precise calculation of many problems having a statistical nature, which would be rather difficult to be solved analytically. For instance, the analysis of the radiation interaction and propagation inside different materials with any arbitrary shape is often performed with MC calculations in Medical Physics and in Nuclear Reactor Physics[3, 4]. In the study of the nuclear properties, the MC calculation has been used in the simulation of the intra-nuclear cascade process following the nuclear interaction with high energy particles, and it has shown to be the more precise method for calculating this process and its characteristics, such as the multiplicity of secondary particles, the secondaries spectra, the residual nucleus formations, etc[5-7].

In this work we develop an OO program which performs a MC calculation for the nuclear evaporation taking place in the residual nucleus formed at the end of the intra-nuclear cascade process. This problem presents all the characteristics described above for an OO algorithm: its is intrinsically statistical, since the competition between evaporation and fission depends on the respective probabilities, and the particles escaping from the nucleus are chosen randomly, according to their escaping probabilities[8]. Also, the process

depends on specific nuclear parameters that can be defined for each nucleus, and the modification of these parameters also affects the nuclear characteristics.

The paper is organized as follows: in section II we describe the nuclear evaporation-fission process, in section III we delineate the OO algorithm developed in the Java programming language, in section IV we show some results obtained with this algorithm, and in section V we present our conclusions.

II Theory: The Nuclear Evaporation-Fission process

The intermediate and high energy nuclear reactions are well described by a two-step process in which the first step is a fast intra-nuclear cascade which transfers most of the incident particle energy to a few nucleons inside the target nucleus[7]. During this process, some particles may escape from the nucleus carrying out part of the initial energy. After a few steps, the number of participating nucleons increases, and the energy of each one is not enough to escape from the nucleus. At this point, an excited residual nucleus is formed, and the excitation energy is uniformly distributed among all its nucleons. The residual nucleus de-excitation mechanism is the evaporation-fission competition, in which some particles escape from the nucleus and/or the nuclear fission takes place[8,9]. The entire process is governed by chance, and just the relative probabilities are used to evaluate the desired physical characteristics in the reaction. During the nuclear evaporation, the k-particle emission probability relative to j-particle emission, according to the Weisskopf's statistical model[8], is

$$\frac{\Gamma_k}{\Gamma_j} = \left(\frac{\gamma_k}{\gamma_j}\right)\left(\frac{E_k^*}{E_j^*}\right)\left(\frac{a_j}{a_k}\right)\exp\left\{2\left[(a_k E_k^*)^{\frac{1}{2}} - \left(a_j E_j^*\right)^{\frac{1}{2}}\right]\right\}. \quad (1)$$

The level density parameter for neutron emission is[10]

$$a_n = 0.134A - 1.21\cdot 10^{-4}A^2 MeV^{-1}, \quad (2)$$

and for all other particles emission this quantity is related to a_n by

$$a_j = r_j a_n, \quad (3)$$

where r_j is a dimensionless constant.

For proton emission we get

$$\frac{\Gamma_p}{\Gamma_n} = \left(\frac{E_p^*}{E_n^*}\right) \exp\left\{2\left(a_n\right)^{\frac{1}{2}}\left[\left(r_p E_p^*\right)^{\frac{1}{2}} - \left(E_n^*\right)^{\frac{1}{2}}\right]\right\}, \tag{4}$$

and for alpha particle emission

$$\frac{\Gamma_\alpha}{\Gamma_n} = \left(\frac{2E_\alpha^*}{E_n^*}\right) \exp\left\{2\left(a_n\right)^{\frac{1}{2}}\left[\left(r_\alpha E_\alpha^*\right)^{\frac{1}{2}} - \left(E_n^*\right)^{\frac{1}{2}}\right]\right\}. \tag{5}$$

The Coulomb potential[9] for proton is

$$V_p = C\frac{\left[K_p\left(Z-1\right)e^2\right]}{\left[r_0\left(A-1\right)^{\frac{1}{3}} + R_p\right]}, \tag{6}$$

and for alpha particle it is

$$V_\alpha = C\frac{\left[2K_\alpha\left(Z-2\right)e^2\right]}{\left[r_0\left(A-4\right)^{\frac{1}{3}} + R_\alpha\right]}, \tag{7}$$

where $K_p = 0.70$ and $K_\alpha = 0.83$ are the Coulomb barrier penetrabilities for protons and alpha particles, respectively, $R_p = 1.14fm$ is the proton radius, $R_\alpha = 2.16fm$ is the alpha particle radius, $r_0 = 1.2fm$. Also,

$$C = 1 - \frac{E^*}{B} \tag{8}$$

is the charged-particle Coulomb barrier correction due to the nuclear temperature [9], with B being the nuclear binding energy. In addition, according to Ref. [13], we use $r_p = r_\alpha = 1$.

Using the fission width from the liquid drop model for fission by Bohr and Wheeler[11], and the neutron emission width from Weisskopf[8], we get[12]

$$\frac{\Gamma_f}{\Gamma_n} = K_f \exp\left\{2\left[\left(a_f E_f^*\right)^{\frac{1}{2}} - \left(a_n E_n^*\right)^{\frac{1}{2}}\right]\right\}, \tag{9}$$

where

$$K_f = K_0 a_n \frac{\left[2\left(a_f E_f^*\right)^{\frac{1}{2}} - 1\right]}{\left(4A^{\frac{2}{3}} a_f E_n^*\right)}, \tag{10}$$

and

$$E_f^* = E^* - B_f, \tag{11}$$

with $K_0 = 14.39MeV$. Here, B_f is the fission barrier height discussed below.

The particle separation energy can be calculated directly from its definition using the Nuclear Mass Formula[15], as we do for the proton and alpha-particle emission, respectively The fission barrier is calculated by[14]

$$B_p = m_p + m(A-1, Z-1) - m(A, Z), \tag{12}$$

and

$$B_\alpha = m_\alpha + m(A-4, Z-2) - m(A, Z), \tag{13}$$

where m_p is the proton mass, m_α is the alpha particle mass, and $m(A, Z)$ is the nuclear mass calculated with the parameters from Ref. [15].

However, for the nuclear fission barrier and for the neutron separation energy there are empirical formulae that give more precise values[14], as, respectively,

$$B_f = C(0.22(A-Z) - 1.40Z + 101.5)MeV; \tag{14}$$

and

$$B_n = (-0.16(A-Z) + 0.25Z + 5.6)MeV. \tag{15}$$

The present Monte Carlo code for Evaporation-Fission (MCEF) calculates, at each step i of the evaporation process, the nuclear fission probability, F_i ,defined as

$$F_i = \frac{\left(\frac{\Gamma_f}{\Gamma_n}\right)_i}{1 + \left(\frac{\Gamma_f}{\Gamma_n}\right)_i + \left(\frac{\Gamma_p}{\Gamma_n}\right)_i + \left(\frac{\Gamma_\alpha}{\Gamma_n}\right)_i}. \tag{16}$$

Then, the particle that will evaporate (neutron, proton or alpha particle) is chosen randomly, according to their relative branching ratio. Once one of these particles is chosen , the mass and atomic numbers are recalculated by

$$A_{i+1} = A_i - \Delta A_i, \tag{17}$$

and

$$Z_{i+1} = Z_i - \Delta Z_i, \tag{18}$$

where ΔA_i, and ΔZ_i are, respectively, the mass and atomic numbers of the ejected particle at the i^{th} step in the evaporation. Also, the nuclear excitation energy is modified according to the expression

$$E^*_{i+1} = E^*_i - B_i - T_i, \tag{19}$$

where B_i and T_i are the separation and the asymptotic kinetic energies of the particle being ejected, respectively. For neutrons, $T = 2MeV$, for protons $T = V_{p,}$and for alpha particles $T = V_\alpha$.

Expression (19) ensures that the nuclear excitation energy will be, at each step in the evaporation chain, smaller than in the previous step. This process continues until the excitation energy available in the nucleus is not enough to emit any of the possible evaporating particles. At this point the evaporation stops, and we can calculate the nuclear fissility by the expression

$$W = \sum_i \left[\prod_{j=0}^{i-1} (1 - F_j) \right] F_i. \tag{20}$$

III Algorithm: The MCEF code

The code organization is based on two objects: the Nucleus, instantiated by the Nucleus class, and the MCEF object, instantiated by the MCEF class.

- Nucleus: is the class where all the nuclear characteristics are calculated and determined. The only way to access its internal variables is through the constructor or through its methods.

- MCEF: is the class which performs the Monte Carlo calculation. It creates an instance of the Nucleus class and modifies its properties according to the evaporation-fission model described in the last section. When a particle evaporates, a new instance of the nucleus class is created with the new values for A, Z and E calculated from our model, and the previous Nucleus object is destroyed. By accessing the fission probability of each nucleus in the evaporation chain, this routine also calculates the fissility of the initial nucleus.

In Figure 1 we show a flow chart of the algorithm, with a short description of each class and the variable through which they communicate with each other. A more comprehensive explanation is given below.

A. The Nucleus class

This class instantiates the Nucleus object through its constructor

```
public Nucleus(int A, int Z, double E){
    this.A=A;
    this.Z=Z;
    this.E=E;
    this.Ex=Ex;
} ,
```

where the input parameters are the nuclear mass number, A, the atomic number, Z, and the nuclear excitation energy, E.

The other nuclear characteristics described in section II are calculated by its methods. Several methods allow to retrieve the resulting value for each nuclear parameter. Below we give an example of such methods for the nuclear fission barrier, Bf. First, Bf is calculated in the method calcBf(), and then it can be retrieved by the method getBf():

```
private void calcBf(){
    Bf=0.22*(A-Z)-1.40*Z+101.5;
    Bf=Bf*(1-E/this.getB());
}

public double getBf(){
    if (Bf==0) this.calcBf();
    return Bf;
}.
```

Note that the method calcBf() needs to retrieve one of the nuclear parameters, the nuclear binding energy B, which is done by the method getB(). In the method getBf(), we first check if the value for the Bf parameter is equal to the initialization value, Bf=0, in order to calculate each parameter only once, thus improving the code performance.

B. The MCEF class

This class instantiates the MCEF object through its constructor

```
public MCEF(int A, int Z, double E){
    this.E=E;
    this.A=A;
    this.Z=Z;
    n=new Nucleus(A,Z,E);
    evap=false;
    W=0;
    nneutrons=0;
    nprotons=0;
    nalphas=0;
} ,
```

where the input parameters are the nuclear mass number, A, the atomic number, Z, and the excitation energy, E, for the initial nucleus in the evaporation process. It instantiates an object Nucleus, n, and a random number generator, r.

This is the steering object for the evaporation process performed by the method Evaporation(), which is the core of the Monte Carlo calculation, and is shown below.

```
private void Evaporation(){
    double Ex=E;
    W=0;
    NF=1;
    double F;
    double rand;
    double Gf,Gp,Ga,Gt;
    while (Ex>0){
        F=this.Probfis();
        W=W+NF*F;
        NF=NF*(1-F);
        Gp=n.getGammap();
        Ga=n.getGammaa();
```

```
            Gt=1+Gp+Ga;
            rand=r.nextDouble();
            if(rand<Gp/Gt){
                A=A-1;
                Z=Z-1;
                Ex=Ex-n.getBp()-n.getVp();
                nprotons=nprotons+NF;
            }
            else{
                if(rand<(Ga+Gp)/Gt){
                    A=A-4;
                    Z=Z-2;
                    Ex=Ex-n.getBa()-n.getVa();
                    nalphas=nalphas+NF;
                }
                else{
                    A=A-1;
                    Z=Z;
                    Ex=Ex-n.getBn()-2.0;
                    nneutrons=nneutrons+NF;
                }
            }
            n=new Nucleus(A,Z,Ex);
        }
        evap=true;
    }
```

Here, the evaporating particle is statistically chosen, according to their respective escaping probabilities, through the random number returned by a static instance of the Random class available in the java.util class to the variable rand, which is compared with the particles emission probability, calculated as the ratio G_i/Gt, where $Gt = 1 + Gp + Ga$

and the index i represents the the evaporanting particle ($i = n$, p or α). The nuclear fissility is also obtained using the fission probability values calculated for each step of the evaporation chain by the method Probfis().

```
private double Probfis(){
    return n.getGammaf()/(1+n.getGammaf()+n.getGammap()+n.getGammaa());
}
```

The boolean variable evap is used to avoid the repetition of the entire Monte Carlo calculation when any variable is needed. If evap has value false, the calculation is performed, otherwise the variable value, which has already been calculated, is returned. An example is in the method which returns the fissility variable, W:

```
public double getFissility(){
    if (evap==false) this.Evaporation();
    return W;
}
```

At each step, the values of A, Z and E are modified according to the evaporated particle characteristics, and the Monte Carlo calculation continues until the nuclear excitation energy is not enough for the evaporation of any possible evaporating particle.

C. Input variables

The input variables are entered through the command line. The variables are:

- A - the mass number;
- Z - the atomic number;
- E - the nuclear excitation energy in MeV;
- stat - the number of iterations for the Monte Carlo Calculation.

D. Output variables

The output variables are displayed at the screen. The variables are:

- W - the calculated mean value for the nuclear fissility;
- Nn - the calculated mean value for the number of neutrons escaping from the nucleus;
- Np - the calculated mean value for the number of protons escaping from the nucleus;
- Na - the calculated mean value for the number of alpha particles escaping from the nucleus;

IV Test Run

Here we show an example of the code output. The input variables are A = 232, Z = 90, E = 900 (MeV) and stat = 500. The input is through the command line java Mcef 232 90 900 500.

The output displayed on the screen is:

- Fissility = 0.75
- Number of Neutrons = 38.73
- Number of Protons = 7,77
- Number of Alphas = 24,22.

We applied this code to calculate the photofissility of the actinide nuclei ^{232}Th, ^{238}U and ^{239}Np at the intermediate photon energy range. The results for their absolute fissility, already published in reference [16], are shown in Figure 2a.

Using the calculated fissility and the experimental photofissility data for those nuclei[16], we obtained the total photoabsorption cross section at intermediate energies. The results are shown in Figure 2b, and compared with the so-called universal curve for the bound nucleon photoabsorption cross section[17], where we observe a good agreement between the universal curve (full lines) and our results. With this work, we solved the long standing problem of the actinide nuclei fissility saturation at values below 100%. For more details, see reference[16].

V Conclusions

We developed an object oriented code written in the Java programming language which performs a Monte Carlo calculation of the evaporation-fission process that happens during many nuclear reactions. The main features of the code are its readability and reusability. Even using the Java language, it presents a good performance during the Monte Carlo calculation. We have shown examples of possible applications, including a recent analisys of some heavy nuclei photofissility[16].

VI ACKNOWLEDGMENTS

This work is supported by the Brazilian agencies FAPESP and CNPq.

References

[1] J. Qiang, R.D. Ryne and S. Habib, Comp. Phys. Comm. 133, 18-33 (2000).

[2] I. Horton in Beginning Java 2, Ed. Wrox Press (1999).

[3] T.D. Solberg et al., Radiochim. Acta 89, 337 (2001).

[4] E. L. Redmond and J. M. Ryskamp, Nucl. Technol. 95, 272 (1991).

[5] V. S. Barashenkov et al,; Nucl. Phys. A231, 462 (1974).

[6] H. W. Bertini; Phys. Rev. 131, 1801 (1963).

[7] M. Gonçalves et al.; Phys. Lett B406, 1 (1997).

[8] V. F. Weisskopf, Phys. Rev. 52, 295 (1937).

[9] O. A. P. Tavares and M. L. Terranova, Z. Phys. A: Hadr. and Nucl. 343, 407 (1992).

[10] A. S. Iljinov, E. A. Cherepanov, and S. E. Chigrinov, Yad. Fiz. 32, 322 (1980) [Sov. J. Nucl. Phys. 32, 166 (1980)].

[11] N. Bohr and J. A. Wheeler, Phys. Rev. 56, 426 (1939).

[12] R. Vandenbosch and J. R. Huizenga, Nuclear Fission, (1^{st}ed., New York Academic Press, 1973) p227.

[13] K. J. LeCouteur, Proc. Phys. Soc. Lond., Ser. A63, 259 (1950).

[14] C. Guaraldo et al., Nuovo Cimento. 103A, 607 (1990).

[15] E. Segrè, Nuclei and Particles, (3^{rd}ed., W. A. Benjamin, INC, 1965) p215.

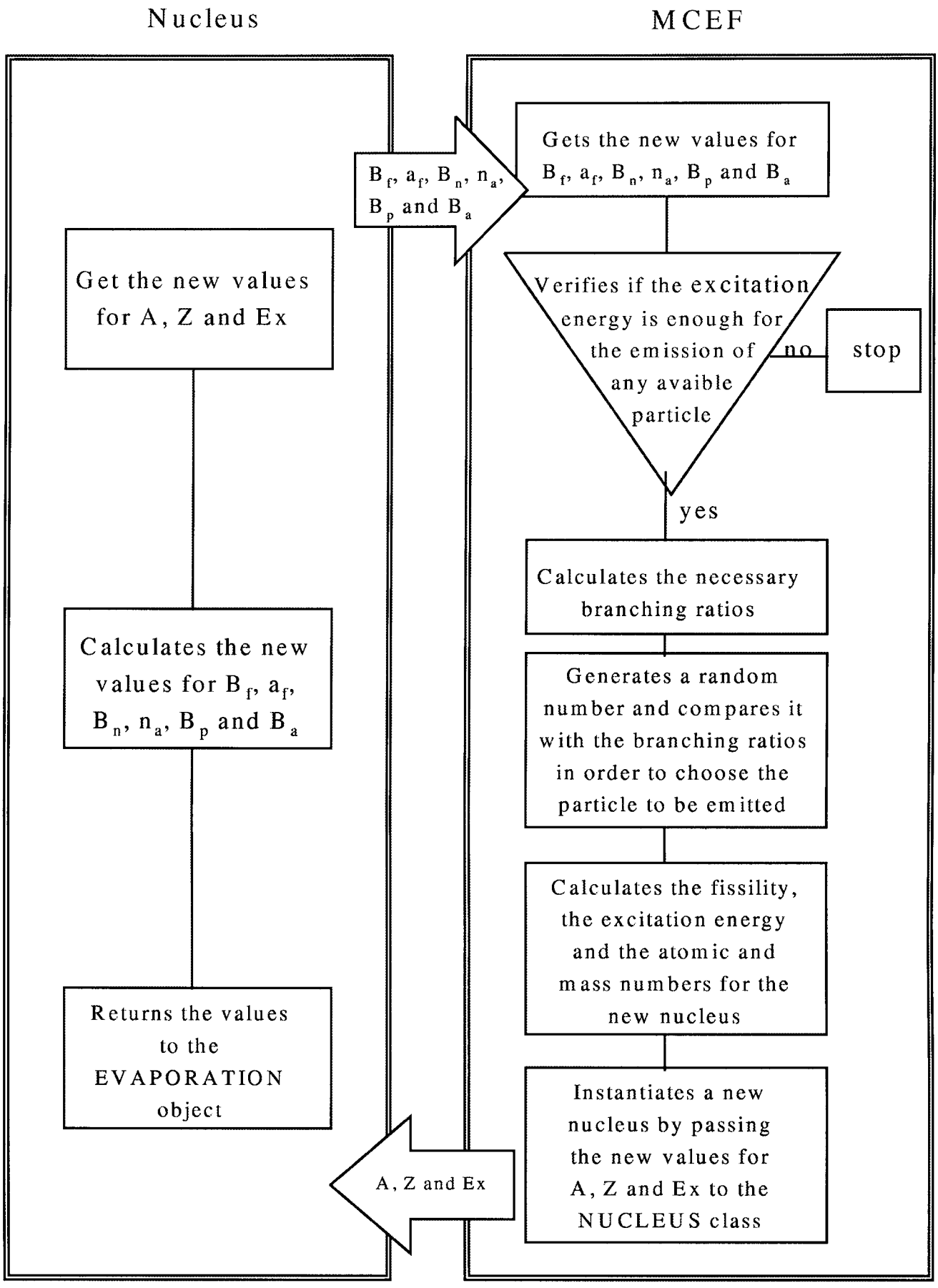
[16] A. Deppman et al., Phys. Rev. Lett, 87 (2001) 182701.

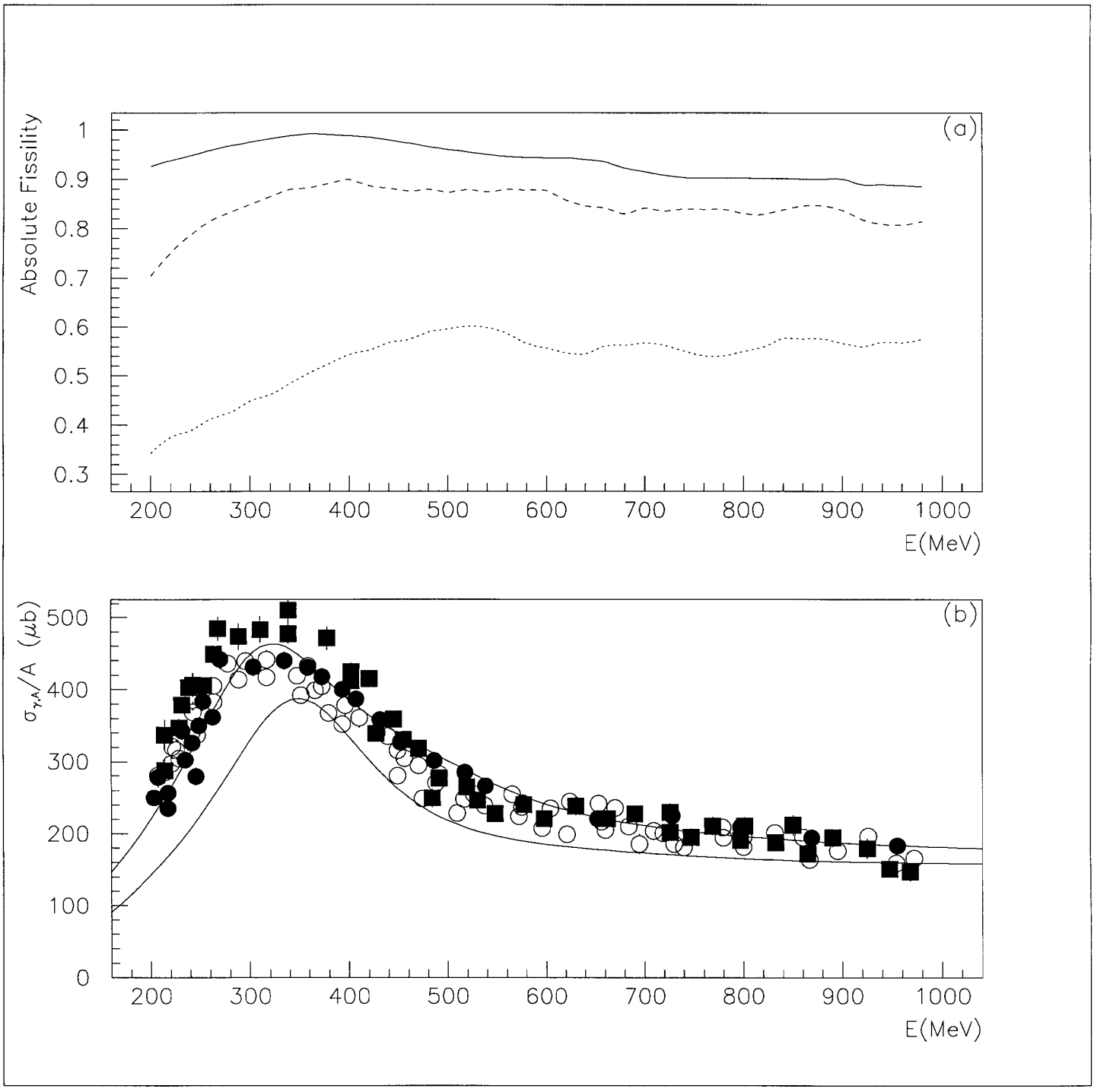
[17] A. Deppman et al., Il Nuovo Cimento 111A, 1299 (1998).

VII Figure Captions

Figure 1: Schematic view of the algorithm. The NUCLEUS and MCEF classes are shown, with a brief description of their functions and the variable values they exchange during the run.

Figure 2:(a) The calculated nuclear fissility as a function of the incident photon energy for ^{237}Np (full line), ^{238}U (dashed line) and ^{232}Th (dotted line). (b) The bound nucleon photoabsorption cross section (see text), as a function of the incident photon energy, for ^{237}Np (full circles), ^{238}U (open circles) and ^{232}Th (full squares).The full lines represent the upper and lower limits for the bound nucleon photoabsorption cross section, as can be deduced from the data reported in [26].


Nucleus
MCEF
Gets the new values for B_f, a_f, B_n, n_a, B_p and B_a
B_f, a_f, B_n, n_a, B_p and B_a
Get the new values for A, Z and Ex
Verifies if the excitation energy is enough for the emission of any avaible particle
no
stop
yes
Calculates the necessary branching ratios
Calculates the new values for B_f, a_f, B_n, n_a, B_p and B_a
Generates a random number and compares it with the branching ratios in order to choose the particle to be emitted
Calculates the fissility, the excitation energy and the atomic and mass numbers for the new nucleus
Instantiates a new nucleus by passing the new values for A, Z and Ex to the NUCLEUS class
A, Z and Ex
Returns the values to the EVAPORATION object


Absolute Fissility
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
(a)
200
300
400
500
600
700
800
900
1000
E(MeV)
$\sigma_{\gamma,A}/A$ (μb)
500
400
300
200
100
0
(b)
200
300
400
500
600
700
800
900
1000
E(MeV)