Simulation of Critical Mass

Of a Fissile Material Joshua Gore 19/9/14

# Aim

The aim of this investigation is to simulate nuclear fission in order to find the critical mass of a fissile material. There are many factors that influence critical mass. This investigation will be examining the effect the size of a cube of fissile material has on its critical mass. This will be done with the help of a computer program using the Monte Python technique, with a cubic block of as the theoretical fissile material.

The simulation will compare mass to the survival fraction of neutrons over a period of time. As the density is held constant the size of the block is proportional to the mass.

# Introduction and Background Theory

In order to understand and therefore simulate factors that influence nuclear fission, an understanding of nuclear fission is essential.

## What is Nuclear Fission?

Nuclear fission occurs when an atomic nucleus subdivides into two approximately equal nuclei. The mass combined mass of these two nuclei is less that of the original nucleus – this is due to the release of a large amount of energy. The energy is related to the mass defect by were m is mass defect and c is the speed of light.[[1]](#footnote-1)

Figure 1 shows a spontaneous nuclear fission – this occurs due to the instability of a nucleus. This releases neutrons, which can cause an induced nuclear fission as shown in figure 3. Figure 2 shows how this occurs.

|  |  |
| --- | --- |
| spontaneous fission  Figure Spontaneous Nuclear Fission[[2]](#footnote-2) | Sequence of events in the fission of a uranium nucleus by a neutron.  Figure how induced fission occurs[[3]](#footnote-3) |
| neutron causing fission  Figure Induced Nuclear Fission[[4]](#footnote-4) |

## Chain reaction

When a nucleus undergoes fission two or three neutrons are emitted. These can then collide with other nuclei and cause them to undergo fission, thus causing a chain reaction. Figure 4 shows a typical chain reaction: . Other fissions of include and .

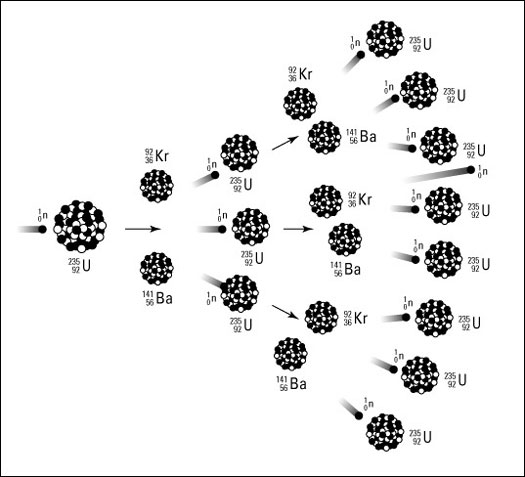


Figure Chain Reaction resulting from Nuclear Fission[[5]](#footnote-5)

## What is Critical Mass?

The critical mass of a fissile material is the smallest amount required to maintain a chain reaction. Not every neutron produced by a fission will cause another fission – some are lost. If neutrons are lost faster than they are created, the mass is subcritical.[[6]](#footnote-6) This can be represented by the survival fraction , where is the number of fissions induced by *N* fissions in the previous generation. Starting with *N* fissions, there will be fissions in the next generation, in the following generation, and in the *n*th generation. A mass for which *f* = 1 is at the critical mass level, and a mass for which *f* >1 is at a supercritical mass level.[[7]](#footnote-7)

## Factors that affect Critical Mass

* Factors that affect critical mass include:
* Enrichment-percentage of atoms that are fissile isotopes
* Shape - The larger the mass to surface area ratio the lower the critical mass, therefore a sphere provides the lowest critical mass.
* Nuclear properties of the fissile – the nuclear cross-section.[[8]](#footnote-8)

## The Monte Carlo Method

The Monte Carlo method was first developed in 1945 in order to simulate nuclear fission chain reactions – the research that lead to Hiroshima.[[9]](#footnote-9) It is a form of statistical sampling – *“Monte Carlo simulation enables us to model situations that present uncertainty and play them out thousands of times on a computer.”[[10]](#footnote-10)* To apply this to fission, the computer would start a fissionable mass, and an initial distribution of atoms in space moving at various velocities. This random These would then be tracked in space – decisions about absorption, escape, and fission being made with random number generators, governed by statistical probabilities.

The model would be run for a number of different starting configurations, thus allowing statistical models to be build. This method can be used to solve many scientific, engineering, and business problems – it is ideal for problems involving random processes.[[11]](#footnote-11)

# Procedure

The following method of simulation nuclear fission and calculating critical mass is based on *Ehrlich, 2002*[[12]](#footnote-12).

The programming language Python 3 will be used, and simulation will be based on *m356p1b.bas* *CRITICAL MASS FOR A FISSION CHAIN REACTION, 1989.[[13]](#footnote-13)*

The purpose of this investigation is to simulate critical mass with the use of technology. This will be done by simulation the fission that occurs in a cubeof a specified mass and shape. The simulation will calculate the *f* value for the given input, by counting the number of induced fissions for a large number of simulated random fissions. The mass of the cubewill be varied, providing a series of values for *f*. The point where the *f* =1 for the model of these points provides the critical mass for the given conditions.

## Requirements

* Some knowledge of the basic and python programming languages
* Working python development environment
* Spreadsheet application to analyse data

## Steps

1. The basic program will be used as a basis for the writing of a python fission simulation program
2. The mass of the block of , ratio of length to thickness, and the number of randomly generated fissions (*N*) will be input to the program, in order to calculate the survival fraction.
3. Step 2 will be repeated for a range of masses.
4. The *f* values obtained from steps 2 and 3 will be graphed and modelled.
5. The model will be used to find the mass when *f* = 1: this is the critical mass.

## Calculation of Critical Mass – Step 2 (Implemented simulation v1.1)

1. Choose the location of the nucleus undergoing fission. This will be a random point with the co-ordinates x0, y0, z0, lying within the boundaries of a block of uranium.

*To simplify calculations the block will have dimensions a \* a \* b, and only the ratio of height to width will be input*

1. Simulate the neutrons caused by the fission: Direction is specified by two random angles, the polar angle θ and the azimuth φ
2. Calculate endpoint of the neutrons, and test whether they are inside the block – neutrons travel a random distance between 0 and 1 units

The endpoint of the neutron has the co-ordinates x1, y1, z1, found by

*It is assumed that neutrons travel any distance between 0 and 1 units with equal probability of hitting another nucleus*

1. Whether the neutrons cause further fission depends on whether they are inside block, based on the co-ordinates found in step 4. Work out whether 0, 1, or 2 neutrons remain in the block, and update *Nin*.
2. *f* is found by *Nin/N*, and output to user

## Changes to procedure

It was decided improve Simulation 1.1, extending the code to simulate any number of generation, allowing for a much more accurate picture of the various reactions that take place. This is accomplished by looping the code, each time updating N to Nin. The user now inputs:

* Ratio of width to height (for this investigation the block of Uranium will be cubic)
* Initial number of randomly generated fissions
* Lower and Upper limit of mass values
* Step of mass values (the program starts at mass = lower limit and repeats until mass = upper limit, incrementing mass by step value)
* Number of generations to simulate for each mass

Three versions of this simulation where created: one simulates fission over time for a single mass value (outputting the survival fraction and number of nucleons), the second the end number of nucleons for a range of mass values, and the third generates both of these (for 3 variable analysis). These three programs are needed to analyse the effect of the mass of the cube over a number of generations, thus providing a more accurate model of the way the size of cube of fissile material effects its critical mass.

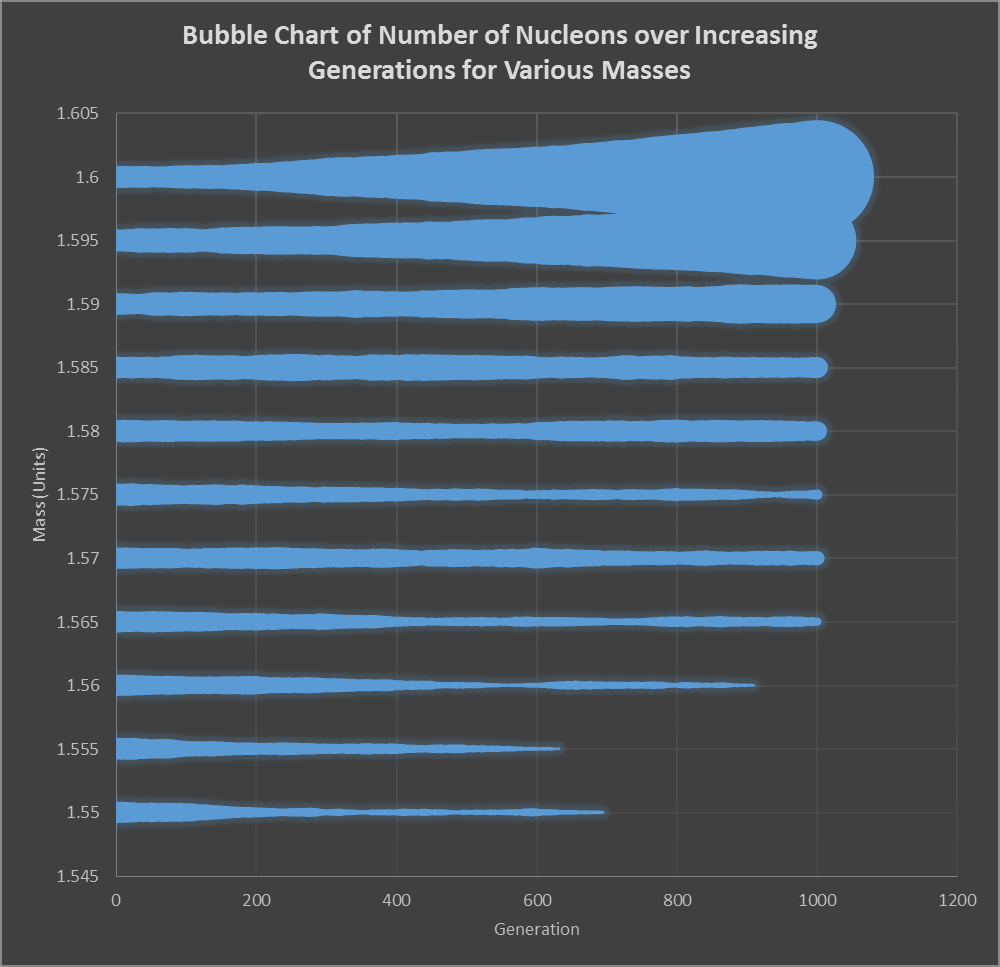
In light of these changes, steps two and three are now as follows:

1. The mass of the block of , ratio of length to thickness(1), number of randomly generated fissions (*N*), the range of mass values, and the number of generations for which fission is to be simulated are input into the simulation.
2. The number of neutrons over a number of generations for a range of masses will be compared: these comparisons can be used to find the approximate critical mass for the cube of

# Results

After some experimentation to find the approximate range for the critical mass, the simulation was run with a step of 0.005, from a mass of 1.55 to a mass of 1.6, for 1000 generations. Each simulated fission chain started with 1000 neutrons.

## Graph and model of results



*The colour scheme of this graph was chosen because it helps to visualise data density. The size of the “bubbles” shows the number of neutron at each generation.*

This chart shows a fairly constant number of nucleons for the mass ranges from 1.57 to1.585. It also shows the random nature of the processes – the survival fraction and therefore the critical mass varies from generation to generation. Over a number of generations this can have a significant effect.

Therefore, according to the simulation, the critical mass of a cube of is between 1.57 and 1.585 units.

The number of fissions in a cube with a mass less than this will decrease every generation, the number of fissions in a cube with a greater mass will increase.

## Graphs of Subcritical, Critical, and Supercritical masses

The below graph shows the number of neutrons over 5000 generations for a fissile cube of mass 1.585 – in the critical mass range. It shows the significant variance in the number of neutrons from generation to generation.

The below graph shows another simulation run with the same parameters, but over a period of 10000 generations – the number of neutrons stays relatively stable until around 5000, and then suddenly increases. Other simulations run with the same parameters suddenly decreased to 0. This serves to show the random nature of critical mass – it is impossible to predict with certainty the exact critical mass over a large number of generations with the developed simulation, if at all.

The graph of a subcritical mass shows an exponential decrease in the number of neutrons. On these larger scales, and for values not so close to the super-critical mass, the values appear much more predictable.

|  |  |
| --- | --- |
|  |  |

# Analysis

## Simulation Limitations

The simulation used in this investigation is limited in a number of ways. It is only simulating the decay of producing 2 neutrons, e.g. . The two neutrons produced have the potential to cause 0, 1, or 2 further fission reactions. There are several reasons why each nucleus might not cause another fission, however the simulation only models one: if the neutron leaves the fissile mass it will not cause further fission.

Herein lies another limitation: the distance travelled by the neutron is calculated as a random number between 0 and 1, with an equal probability of the number being anywhere within that range. This is unrealistic – there should be a statistical bias towards certain travelled distances.

The direction of movement of two neutrons from each other was completely random: in a real fission their direction is related (conservation of momentum/energy).

A significant limitation in this simulation is that there are no units for mass and distance: these would be difficult to implements without simulating many more of the fission variables.

## Improvements

All the above listed limitations are sources of random error in this simulation, so in order to improve accuracy they should be addressed. This would involve further Monty Carlo based algorithms, and the generation of large amounts of data (to obtain good samples of the possibilities for each of the variables).

Given the random nature both of nuclear fission and the Monty Carlo method, it is virtually impossible to “eliminate” random error: every simulated and every real chain reaction will be different

# Conclusion

This investigation has simulated nuclear fission, finding the critical mass of a cube of fissile material using the Monte Python technique. The simulation algorithm developed was of a limited nature – whether a nucleus caused a fission or not was decided simply by whether it ended up inside or outside the fissile cube. By this algorithm the critical mass of a cube of was calculated to be in the range of 1.57 to 1.585 units.

The simulation showed how that a cube with a larger mass, and therefore longer dimensions, will result in a larger survival rate than a smaller cube. It also showed that is not possible to find and exact critical mass value: due to the random nature of fission the survival rate of neutrons for a mass near critical increases and decreases significantly – the number of neutrons can stay reasonably steady for 5000 generations, and then suddenly increase exponentially, or drop to 0 resulting in the chain reaction completely stopping.

# Appendix A: Details of Simulation Software

## V1.1 – working port

### Simulation 1.1 Code:

1. \_\_author\_\_ = 'Joshua Gore'
2. """
3. CRITICAL MASS FOR A FISSION CHAIN REACTION
4. ---------------------17/9/14---------------------
5. This program computes the survival fraction, F,
6. for a rectangular slab of fissionable material. F
7. is defined as the number of induced fissions per
8. spontaneous fission. For values of F greater than
9. 1.0, a chain reaction occurs. The slab has
10. "critical mass" if it has F=1. We assume that two
11. neutrons are emitted per fission.
12. """
14. #initialise
16. **import** math
17. **import** random

20. **def** rand9():

This random number generator is an improvement on that provided by original code

1. r = []
2. **for** k **in** range(0, 9):
3. r.append(random.random())
4. **return** r
6. #Input
7. y = 'y'
8. **while** y == 'y' **or** y == 'Y':
9. **print**("INPUT VALUES FOR THE FOLLOWING: (PRESS RETURN TO ENTER)")
10. M = float(input(" THE MASS OF THE RECTANGULAR SLAB (M): "))
11. S = float(input(" THE RATIO OF LENGTH TO THICKNESS (S): "))
12. N = float(input(" THE NUMBER OF RANDOMLY GENERATED FISSIONS (N): "))
13. **print**("")
15. # Calculate dimensions
16. A = (M \* S) \*\* (1 / 3)
17. B = (M / S \*\* 2) \*\* (1 / 3)
18. L1 = A / 2
19. L2 = A / 2
20. L3 = B / 2
22. """
23. --Set NIN, the number of cases in which an emmited
24. neutron stops within the boundaries of the slab,
25. thus producing an induced fission, equal to zero
26. initially.
27. """
28. NIN = 0
29. N = int(N)
31. **for** i **in** range(0, N):
32. """
33. #For each spontaneous fission we need 9 random
34. #numbers
35. """
37. R = rand9()  # generates 9 random numbers
38. """
39. -- Calculate the X, Y, and Z co-ordinates of the
40. nucleus undergoing spontaneus emmision. They
41. are random numbers inside the bondary of the
42. block.
43. Note: X,Y are in the interval (-A/2,+A/2)
44. Z is in the interval (-B/2,+B/2)
45. """
46. X0 = A \* (R[0] - 0.5)
47. Y0 = A \* (R[1] - 0.5)
48. Z0 = B \* (R[2] - 0.5)
50. #begin loop over each of the two neutrons; 1 at a time
51. K = 1
52. **for** K **in** range(1, 3):
53. #get two angles that define a random direction
54. #in space for the neutron:
55. PHI = 2 \* math.pi \* R[2 \* K + 1]
56. COSTH = 2 \* (R[2 \* K + 2] - 0.5)
57. SINTH = math.sqrt(1 - COSTH \*\* 2)
58. #get distance traveled by the neutron,
59. #assumed to be a random number in the interval (0,1)
60. D = R[K + 6]
61. #calculate the coordinates of the end point
62. #(interaction point) for the neutron
63. X1 = X0 + D \* SINTH \* math.cos(PHI)
64. Y1 = Y0 + D \* SINTH \* math.sin(PHI)
65. Z1 = Z0 + D \* COSTH
66. #if the end point is inside the block,
67. #the neutron produces an induced fission so add 1 to NIN:
68. **if** abs(X1) <= L1 **and** abs(Y1) <= L2 **and** abs(Z1) <= L3:
69. NIN += 1
70. K += 1
71. #output
72. **print**("             MASS =", M)
73. **print**(" LENGTH/THICKNESS =", S)
74. **print**("# RANDOM FISSIONS =", N)
76. #the survival fraction is NIN/N:
77. F = NIN / N
78. **print**("SURVIVAL FRACTION:", F)
79. **print**()
80. y = input("REPEAT? (y): ")

## Simulation 3.0 Code: Multiple Generations

1. #\_\_author\_\_ = 'Joshua Gore'
2. """
3. CRITICAL MASS FOR A FISSION CHAIN REACTION
4. ---------------------18/9/14---------------------
5. This program computes the survival fraction, F,
6. for a rectangular slab of fissionable material. F
7. is defined as the number of induced fissions per
8. spontaneous fission. For values of F greater than
9. 1.0, a chain reaction occurs. The slab has
10. "critical mass" if it has F=1. We assume that two
11. neutrons are emitted per fission.
12. """
14. #initialise
16. **import** math
17. **import** random
18. **from** time **import** gmtime, strftime
19. Print = "G, F, M, N, \n"
20. M = 0.0
21. Gen = 0
23. # generates 9 random numbers
24. **def** rand9():
25. r = []
26. **for** k **in** range(0, 9):
27. r.append(random.random())
28. **return** r
30. #Input
31. **print**("INPUT VALUES FOR THE FOLLOWING: (PRESS RETURN TO ENTER)")
32. S = float(input(" THE RATIO OF LENGTH TO THICKNESS (S): "))
33. NN = float(input(" THE INITIAL NUMBER OF RANDOMLY GENERATED FISSIONS (N): "))
34. G = float(input(" THE NUMBER OF SIMULATED GENERATIONS (G): "))
35. M = float(input(" LOWER LIMIT OF MASS VALUES: "))
36. Upper = float(input(" UPPER LIMIT OF MASS VALUES: "))
37. Step = float(input(" STEP OF MASS VALUES: "))
38. **print**("")
39. UpperP = Upper
40. GP = G
41. NP = NN
42. LP = M
43. #Hack due to float imprecision
44. Upper += 0.0000000000001
45. **while** M <= Upper:
46. N = NN #replace this but stops breaking code
47. Gen = 0
49. # Calculate dimensions
50. A = (M \* S) \*\* (1 / 3)
51. B = (M / S \*\* 2) \*\* (1 / 3)
52. L1 = A / 2
53. L2 = A / 2
54. L3 = B / 2
55. **while** Gen <= G:
56. """
57. --Set NIN, the number of cases in which an emitted
58. neutron stops within the boundaries of the slab,
59. thus producing an induced fission, equal to zero
60. initially.
61. """
62. NIN = 0
63. N = int(N)
65. **for** i **in** range(0, N):
66. """
67. #For each spontaneous fission we need 9 random
68. #numbers
69. """
71. R = rand9()  # generates 9 random numbers
72. """
73. -- Calculate the X, Y, and Z co-ordinates of the
74. nucleus undergoing spontaneous emission. They
75. are random numbers inside the boundary of the
76. block.
77. Note: X,Y are in the interval (-A/2,+A/2)
78. Z is in the interval (-B/2,+B/2)
79. """
80. X0 = A \* (R[0] - 0.5)
81. Y0 = A \* (R[1] - 0.5)
82. Z0 = B \* (R[2] - 0.5)
84. #begin loop over each of the two neutrons; 1 at a time
85. K = 1
86. **for** K **in** range(1, 3):
87. #get two angles that define a random direction
88. #in space for the neutron:
89. PHI = 2 \* math.pi \* R[2 \* K + 1]
90. COSTH = 2 \* (R[2 \* K + 2] - 0.5)
91. SINTH = math.sqrt(1 - COSTH \*\* 2)
92. #get distance traveled by the neutron,
93. #assumed to be a random number in the interval (0,1)
94. D = R[K + 6]
95. #calculate the coordinates of the end point
96. #(interaction point) for the neutron
97. X1 = X0 + D \* SINTH \* math.cos(PHI)
98. Y1 = Y0 + D \* SINTH \* math.sin(PHI)
99. Z1 = Z0 + D \* COSTH
100. #if the end point is inside the block,
101. #the neutron produces an induced fission so add 1 to NIN:
102. **if** abs(X1) <= L1 **and** abs(Y1) <= L2 **and** abs(Z1) <= L3:
103. NIN += 1
104. K += 1
106. #output
107. #if i == 0:
108. #print("             MASS= ", M)
109. #if i == 0:
110. #print(" LENGTH/THICKNESS= ", S)
111. #if i == 0:
112. #print("# RANDOM FISSIONS= ", N)
113. #if i == 0:
114. #print("GEN, SURVIVAL FRACTION, MASS, NUMBER")
116. #the survival fraction is NIN/N:
117. **if** N != 0:
118. F = NIN / N
119. **else**:
120. F = 0
121. **print**(Gen, F, M, N, sep=',')
122. Print = Print + str(Gen) + ', ' + str(F) + ', ' + str(M) + ', ' + str(N)
123. Print += "\n"
124. N = NIN #update number of neutrons should this be here!!!
125. Gen += 1
126. M += Step
127. #print(Print)
128. fname = 'output '
129. fname = fname + '(' + str(Step) + ')' + '(' + str(LP) + ')' + '(' + str(UpperP) + ')' + '(' + str(NP) + ') ' + '(' + str(GP) + ') '
130. fname += strftime("%Y-%m-%d %H-%M-%S", gmtime())
131. fname += ".csv"
132. with open(fname, 'w') as fout:
133. fout.write(Print)

1. nuclear fission, (2014). *Britannica School*, [online] available at: [http://school.eb.com.au/levels/high/article/110414#](http://school.eb.com.au/levels/high/article/110414) [accessed 17 September 2014] nuclear fission, (2014). *Britannica School*, [online] available at: [http://school.eb.com.au/levels/high/article/110414#](http://school.eb.com.au/levels/high/article/110414) [accessed 17 September 2014] [↑](#footnote-ref-1)
2. Ofnuclearenergy.com, (2014). *Nuclear fission*. [online] Available at: http://ofnuclearenergy.com/working/fission-nuclear.html [Accessed 17 Sep. 2014]. [↑](#footnote-ref-2)
3. nuclear fission, (2014). *Britannica School*, [online] available at: [http://school.eb.com.au/levels/high/article/110414#](http://school.eb.com.au/levels/high/article/110414) [accessed 17 September 2014] [↑](#footnote-ref-3)
4. Ofnuclearenergy.com, *op. cit.* [↑](#footnote-ref-4)
5. Dummies.com, (2014). *Nuclear Fission Basics - For Dummies*. [online] Dummies.com. Available at: http://www.dummies.com/how-to/content/nuclear-fission-basics.html [Accessed 16 Sep. 2014]. [↑](#footnote-ref-5)
6. *ibid* [↑](#footnote-ref-6)
7. Ehrlich, R. (2002). *PHYSNET MISN-0-356 SIMULATION OF A CHAIN REACTION.* 1st ed. [pdf] Fairfax: Physics Dept., George Mason Univ. Available at: http://www.physnet.org/modules/pdf\_modules/m356.pdf [Accessed 17 Sep. 2014]. [↑](#footnote-ref-7)
8. Cyberphysics.co.uk, (2014). *Critical Mass*. [online] Available at: http://www.cyberphysics.co.uk/topics/nuclear/critical.htm [Accessed 17 Sep. 2014]. [↑](#footnote-ref-8)
9. Ehrlich, R. (2002). *PHYSNET MISN-0-356 SIMULATION OF A CHAIN REACTION.* 1st ed. [pdf] Fairfax: Physics Dept., George Mason Univ. Available at: http://www.physnet.org/modules/pdf\_modules/m356.pdf [Accessed 17 Sep. 2014]. [↑](#footnote-ref-9)
10. Office.microsoft.com, (2014). *Introduction to Monte Carlo simulation - Excel*. [online] Available at: http://office.microsoft.com/en-au/excel-help/introduction-to-monte-carlo-simulation-HA001111893.aspx#BMsimulate\_discrete [Accessed 18 Sep. 2014]. [↑](#footnote-ref-10)
11. Metropolis, N. (1987). *"The beginning of the Monte Carlo method".* Los Alamos Science (1987 Special Issue dedicated to Stanislaw Ulam). Available at: http://library.lanl.gov/cgi-bin/getfile?00326866.pdf [accessed 19 Sep 2014]. [↑](#footnote-ref-11)
12. Ehrlich, R. (2002). *PHYSNET MISN-0-356 SIMULATION OF A CHAIN REACTION.* 1st ed. [pdf] Fairfax: Physics Dept., George Mason Univ. Available at: http://www.physnet.org/modules/pdf\_modules/m356.pdf [Accessed 17 Sep. 2014]. [↑](#footnote-ref-12)
13. m356p1b.bas CRITICAL MASS FOR A FISSION CHAIN REACTION. (1989). www.physnet.org. [↑](#footnote-ref-13)