Monte Carlo calculation of the flight distance to the first collision

Alan L. Arsen

28 November 2019

Monte Carlo Methods and Simulations
in Nuclear Technology (SH2704)
KTH Royal Institute of Technology

Problem definition

Using a Monte Carlo simulation, calculate the mean distance that the fission neutrons fly until their first collision. Use an infinite system composed of a single fissile nuclide at a reasonable mass density.

Perform a simple sampling simulation to evaluate the aforementioned parameter. Collect at least several thousands of samples and compute the mean distance and the variance of the mean value.

Watt Fission Spectrum

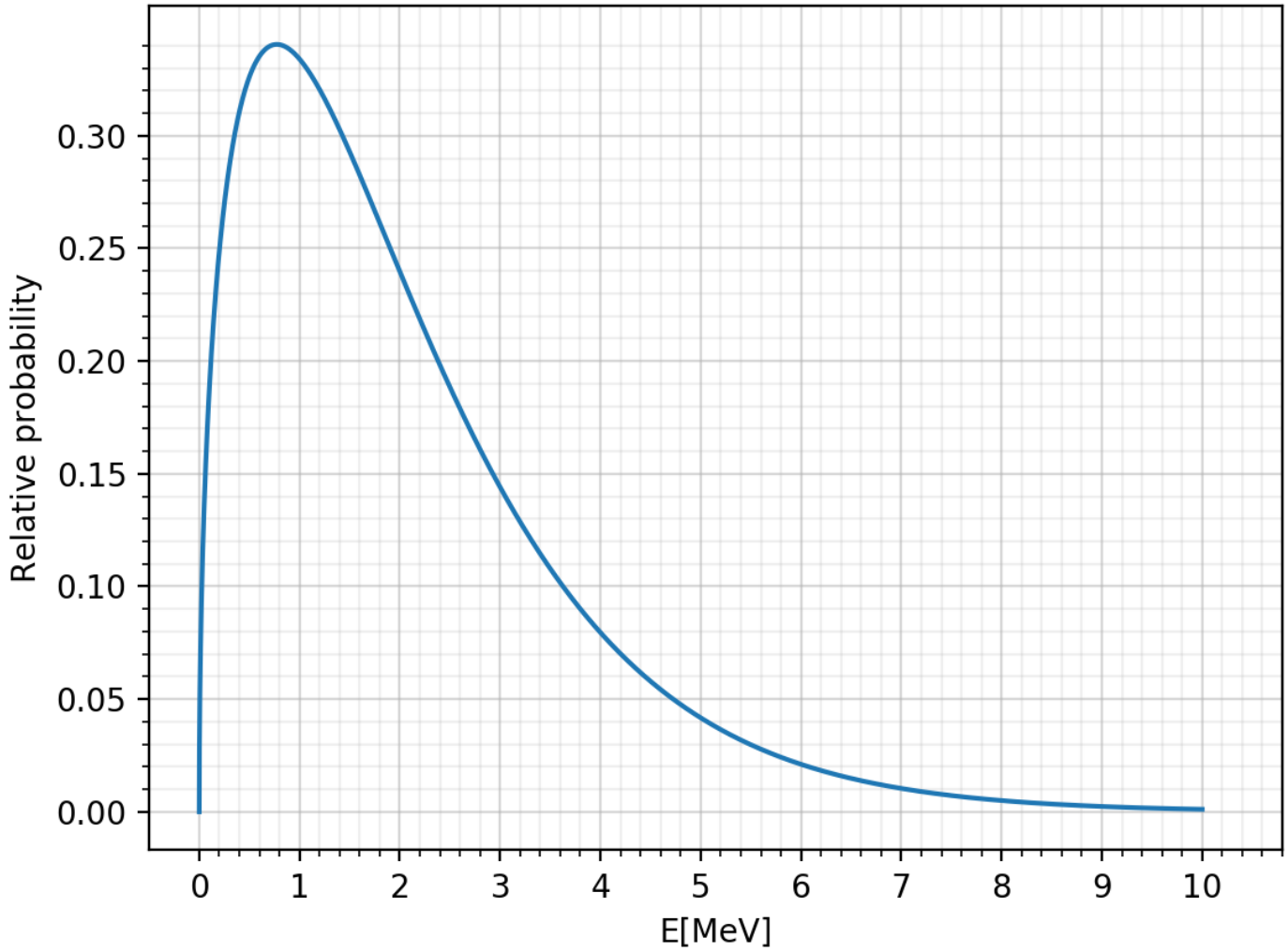
The analytical expression of the Watt Fission Spectrum is:

$$\chi(E) = C\, e^{-E/a} \sinh(\sqrt{bE}),$$

Where a and b are constants that depend on the nuclide and C is the normalization constant. This study was performed using the constants corresponding to U-233 for thermal fissions which are[1]:

$$a = 0.977; b = 2.546.$$

Watt Fission Spectrum



Solution approach

The system was assumed to be made of U-233 with an atomic density given by:

$$N \approx \frac{19.1 \ \frac{g}{cm^3} \ N_a}{233},$$

where N_a is the Avogadro number.

The energy was sampled using acceptance rejection method, sampling the values in a box from 0 to 20 MeV in the x-axis and 0 to $\max[\chi(E)]$ in the y-axis.

Solution approach

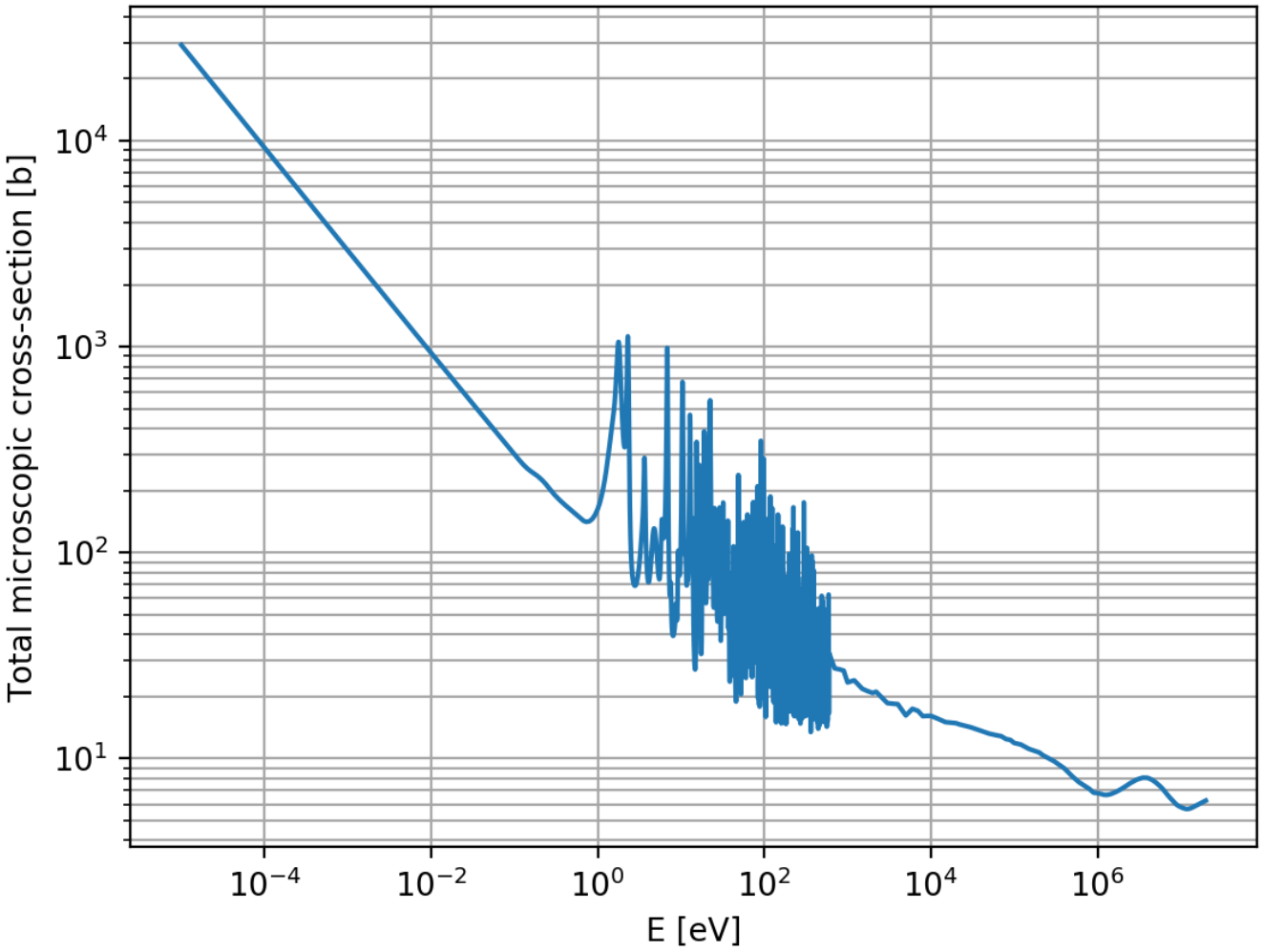
The cross section data was obtained from ENDF/B-VIII.0 library[2]. The cross section for a given energy was obtained using a linear interpolation.

The distance between collisions was sampled using the inverse transform method knowing that the CDF for this process is given by:

$$F_s = 1 - e^{-\Sigma_t s},$$

where Σ_t is the total cross section and s the parameter of interest.

U – 233 total cross-section



Solution implementation: coding

```
# Functions definition
#
watt  = lambda var,paramA,paramB,paramC: paramC * np.exp(-var/paramA) * np.sinh(np.sqrt(paramB*var))
x     = lambda var: var
x2    = lambda var: var**2
funOp = lambda var,paramA,paramB,paramC,fun1,fun2: fun1(var,paramA,paramB,paramC) * fun2(var)
normF = lambda paramA,paramB,paramC,fun: paramC/(quad(fun, 0, np.inf, args=(paramA,paramB,paramC))[0])
probF = lambda delta, sigma: erf(delta/(sigma * np.sqrt(2)))

def findXS(argE, argDataXS):

    i = 0
    while argE > argDataXS[i,0]:
        i += 1
    interp = interp1d(data[i-1:i+1,0], data[i-1:i+1,1])

    return interp(argE)

sortS = lambda argXS, numDensU: -(1/(argXS*numDensU)) * np.log(random.random())
```

Solution implementation: coding

```
def singleSim(argSimNum, argEnergyScalling, argProbScalling, argDataXS,numDensU):

    i = 0
    fcount = 0
    fSumE  = 0
    fSumE2 = 0
    fSumS  = 0
    fSumS2 = 0

    while i<int(argSimNum):
        fcount = fcount + 1
        E = argEnergyScalling * random.random()
        P = argProbScalling * random.random()
        if watt(E,a,b,c) >=  P:

            i = i + 1
            s = sortS(findXS(E*1e6, argDataXS)*1e-24,numDensU)
            fSumE  += E
            fSumE2 += E**2
            fSumS  += s
            fSumS2 += s**2

    return fSumE, fSumE2, fcount, fSumS, fSumS2
```

Solution implementation: coding

```
# Main
#
simNum = 1e5
energyScalling = 20 # MeV
probScalling = watt(maxEdet,a,b,c)

sumE, sumE2, count, sumS, sumS2 = singleSim(simNum, energyScalling, probScalling, data, numDens)

meanE = sumE/simNum
stDvMeanE = np.sqrt((sumE2/simNum - meanE**2)/simNum)
deltaE = 2 * stDvMeanE
pE = probF(deltaE,stDvMeanE) # Confidence interval

meanS = sumS/simNum
stDvMeanS = np.sqrt((sumS2/simNum - meanS**2)/simNum)
deltaS = 2 * stDvMeanS
pS = probF(deltaS,stDvMeanS) # Confidence interval

eff = simNum/count
```

Results

Parameter	Monte Carlo*
Simulation result	2.70 ± 0.02 cm
Confidence interval	0.954
Efficiency of the sampling method**	14.62%
Variance	7E-5 cm^2

*The simulation was performed using 1E5 accepted samples

** Defined as the ratio of accepted samples over the total number of samples

References

[1] X-5 Monte Carlo Team. “MCNP — A General Monte CarloN-Particle Transport Code, Version 5”. Volume I: Overview and Theory. 2003.

[2] https://www.oecd-nea.org/janisweb/book/neutrons/U233/MT1/renderer/12

Thank you for your attention

Tack för er uppmärksamhet