Sample mean, variance and variance of the mean value of the energy distribution of fission neutrons

Alan L. Arsen

21 November 2019

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

Problem definition

Having the probability density function that describes the energy distribution of fission neutrons coming from a specific fissile nuclide, generate at least 1000 samples randomly from this distribution by the acceptance-rejection method, and use these samples to estimate:

1. the mean value of the fission neutron energy,
2. the variance and the standard deviation of the energy of the fission neutrons,
3. confidence intervals for the estimated mean value,
4. the variance of the mean value.

Watt Fission Spectrum

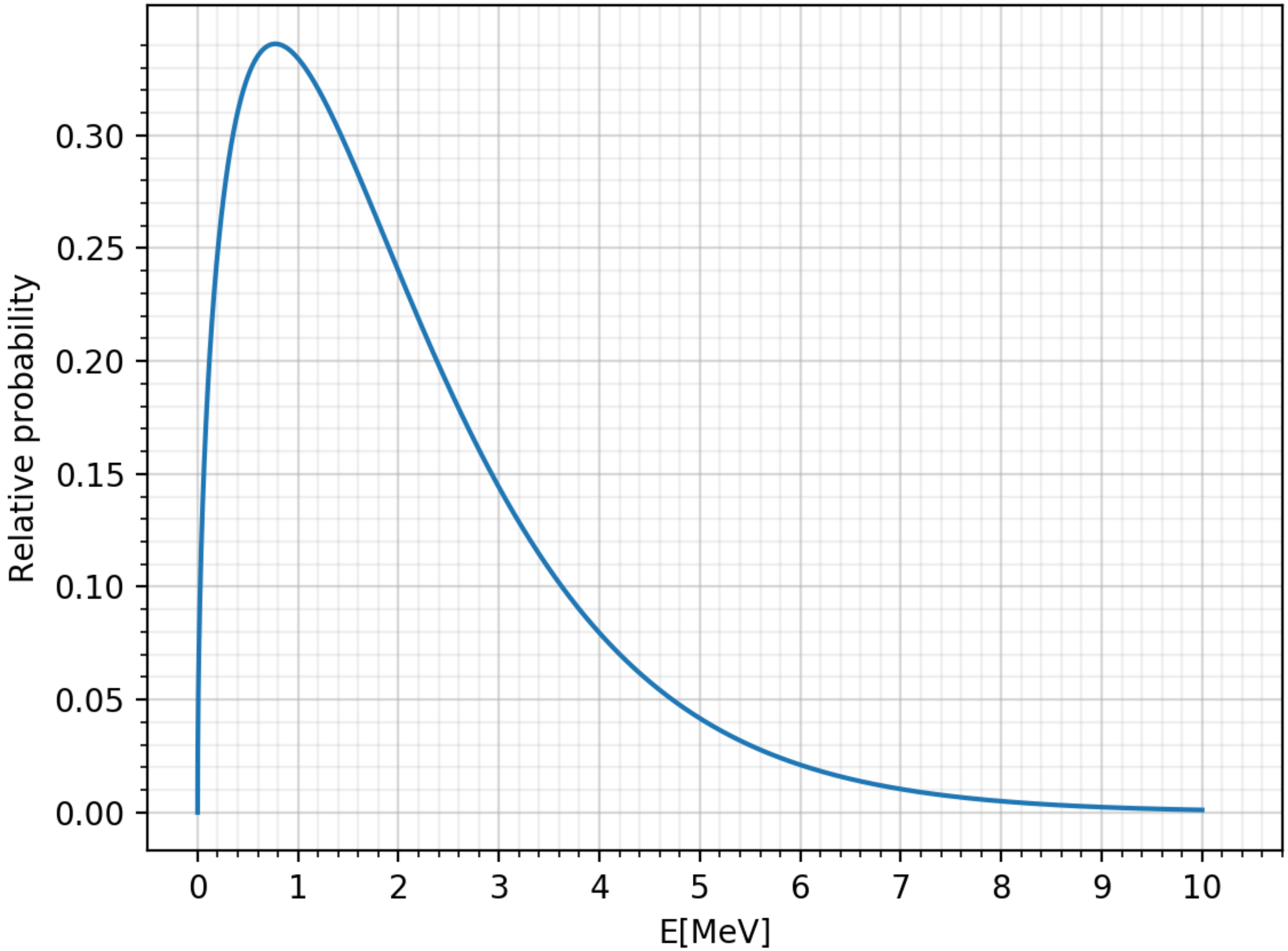
The analytical expression of the Watt Fission Spectrum is:

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

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

1. The mean value of the random variable neutron energy E, can be obtained as:

$$m_E = \frac{1}{n}\sum_{i=1}^{n} e_i \,,$$

where e_i are the sampled values of E and n the total number of simulations.

2. The estimation of the variance of E, can be calculated as:

$$S_E^2 = \frac{1}{n}\sum_{i=1}^{n} e_i^2 - m_E^2 \,,$$

where S_E is the estimation of the standard deviation.

Solution approach

3. The estimation of the variance of m_E, can be calculated as:

$$S_{m_E}^2 = \frac{1}{n} S_E^2.$$

4. The probability P that m_E is inside $[\mathrm{E}(E) - \delta, \mathrm{E}(E) + \delta]$ is:

$$P = \mathrm{erf}\left(\frac{\delta}{\sigma_{m_E}\sqrt{2}}\right),$$

where $\mathrm{erf}(\mathrm{x})$ is the Gauss error function and the standard deviation of m_E, denoted by σ_{m_E}, is estimated as S_{m_E}.

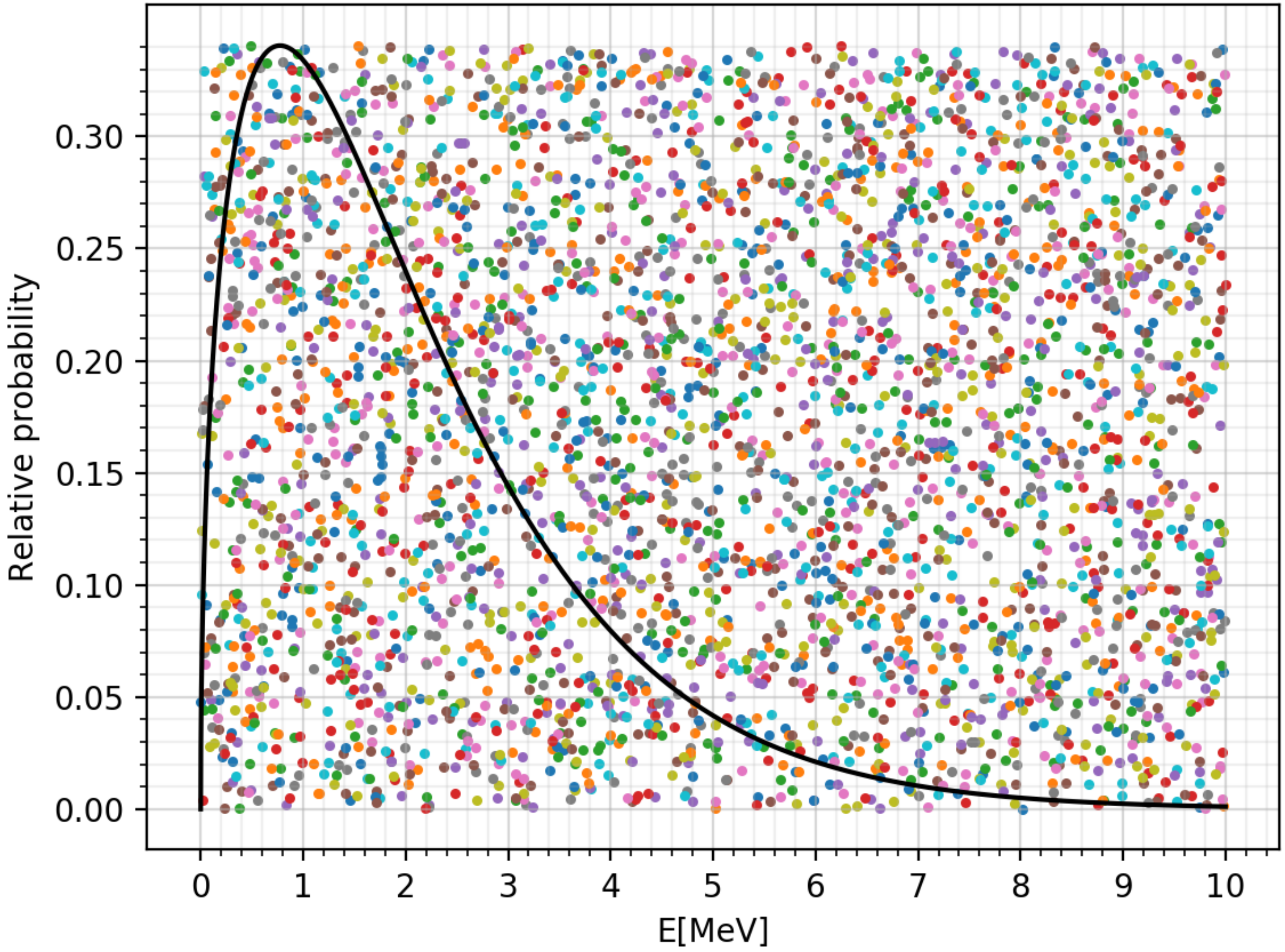
Solution implementation: coding

```
# 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 singleSim(argSimNum, argEnergyScalling, argProbScalling):

    fcount = 0
    i = 0
    fSumE = 0
    fSumE2 = 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

            fSumE = fSumE + E
            fSumE2 = fSumE2 + E**2

    return fSumE, fSumE2, fcount
```

Solution implementation: coding

```
# Main
#
print("\nRunning Monte Carlos simulation...")

simNum = 1e5
energyScalling = 30 # MeV
probScalling = watt(maxEdet,a,b,c)
runNum = 1e3
acceptedN = 0

for i in range(int(runNum)):

    print(i+1)
    sumE, sumE2, count = singleSim(simNum, energyScalling, probScalling)
    meanE = sumE/simNum
    stDvMeanE = np.sqrt((sumE2/simNum - meanE**2)/simNum)
    delta = 2 * stDvMeanE

    if meanE+delta > meanEdet and meanE-delta < meanEdet:
        acceptedN = acceptedN + 1

varE = sumE2/simNum - meanE**2
stDvE = np.sqrt(varE)
varMeanE = varE/simNum
p = probF(delta,stDvMeanE) # Confidence interval
eff = simNum/count
accepPorc = acceptedN/runNum
```

Solution implementation: sampling



Results

Parameter	Monte Carlo*	Deterministic
Simulation result	2.072 ± 0.010 MeV	2.073 MeV
Confidence interval	0.954	-
Efficiency of the sampling method**	9.77%	-
Variance	2.619 MeV^2	2.636 MeV^2
Standard deviation	1.623 MeV	1.624 MeV
Variance of the mean value	2.621E-5 MeV^2	-
Percentage of simulations with the accurate expectation value inside the confidence interval	95.4%	-

*All simulations were 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.

Thank you for your attention

Tack för er uppmärksamhet