

# Programming Techniques for Scientific Simulations

## Exercise 8

### Problem 8.1 Penna Model Implementation

Design and implement a `Population` class that performs all major operations on a population of animals (aging, generation of offsprings, deaths) and combine the classes into a working simulation of the Penna model.

Once the simulation is working test it by setting the simulation parameters to values given in the paper and reproducing its Fig. 1: population number as a function of time. Also plot the distribution of bad genes in a genome at the beginning and at the end of the simulation.

### Problem 8.2 Acceptance-rejection method

Write your own random number distribution based on the acceptance-rejection method. It shall calculate random numbers distributed according to the probability density

$$f(x; p) = \frac{1}{z} \cos^2\left(\frac{x}{p}\right) \exp\left(-\frac{x^2}{2p^2}\right) \quad (1)$$

with the normalisation  $z$  and a parameter  $p$ . Use the `normal_distribution`  $n(x; \mu = 0, \sigma = p)$  as the bounding distribution.<sup>1</sup>

- Create a function object with the parameter  $p$  taken by the constructor.
- Try to fulfill the requirements of existing C++11 random number distributions and make your distribution standard conforming. Check the requirements for random number distributions in the C++ standard.<sup>2</sup>
- Test your random numbers by calculating the mean and the standard deviation<sup>3</sup> which is 0, and  $\sqrt{\frac{e^2-3}{e^2+1}} p \approx 0.723317573 p$ , resp.
- You can also check the acceptance rate of your acceptance-rejection method, which in this specific case is given by

$$p_{\text{accept}}(p) = \frac{\text{number of calls}}{\text{number of attempts}} = \frac{e^2 + 1}{2e^2}. \quad (2)$$

Note, that this acceptance rate is only achieved if the smallest possible constant  $\lambda$  is chosen such that  $f(x) < \lambda n(x)$  for all  $x$ .

<sup>1</sup>[http://en.wikipedia.org/wiki/Normal\\_distribution](http://en.wikipedia.org/wiki/Normal_distribution)

<sup>2</sup><http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2012/n3337.pdf> (page 892ff), <http://en.cppreference.com/w/cpp/concept/RandomNumberDistribution>

<sup>3</sup>[http://en.wikipedia.org/wiki/Standard\\_deviation](http://en.wikipedia.org/wiki/Standard_deviation)

### Problem 8.3 Monte Carlo integration

Calculate the value of  $\pi$  using Monte Carlo integration using `mt19937` (from the C++11 standard library or `Boost.Random`) and `drand48` as your random number generator.

- Draw random numbers and check whether they are within the unit circle. The number of hits divided by the total number of trials gives you an estimate for  $\frac{\pi}{4}$ .
- Calculate the standard error of the mean.<sup>4</sup>
- Calculate the difference of the Monte Carlo estimate for  $\pi$  with the actual value of  $\pi$ . Compare this difference with the standard error of the mean. What do you observe concerning the two random number generators?

---

<sup>4</sup>You may consult [http://en.wikipedia.org/wiki/Standard\\_error\\_\(statistics\)](http://en.wikipedia.org/wiki/Standard_error_(statistics)) for the formula.