Exercise in C++ Programming for CE
ExC++PCE
Winter Term 2014/2015

Mini Project (Monte Carlo Simulation)

General Remarks

StudOn submission

This mini project is an individual project, i.e., each student has to work by herself. *No group-work*! Please note that, as for the StudOn submissions of the questions on the assignment sheets, the successful completion of this mini project is mandatory. Please check the StudOn course[1] regularly for further information about the deadline and the submission of you solutions.

Monte Carlo Simulation

Monte Carlo methods are a class of computational algorithms that rely on repeated random sampling for the computation of the result. They are often used for the simulation of physical and mathematical systems, as well as for random testing. Due to the fact that they rely on random, or rather pseudo-random, numbers these methods are well suited for computer simulations and tend to be used when it is infeasible or impossible to compute an exact result. As already mentioned, Monte Carla methods can be used in many different fields of application; for a bit of history see: `http://jackman.stanford.edu/mcmc/metropolis1.pdf`

Approximation of Pi

The aforementioned Mote Carlo methods can be used to compute (approximate) the value of Pi. For this, consider a circle with radius *r* inside a square of side length *2r (cf. figure 1)*. The approximation uses randomly generated coordinate pairs (points) and evaluates for each point if it lies within or outside the circle. From the properties of a circle we know that its area is defined by $\pi * r^2$, as compared to $4 * r^2$ for the surrounding square. Combined with the probability of the random points to be either inside or outside the circle, we can deduce:

$$\pi \approx \frac{4 * (\#\ of\ points\ inside\ circle)}{(\#\ of\ points\ inside\ square)}$$

This is because the probability of a point to be inside the circle is the ratio of the area of the circle and the area of the square. If a point lies within the circle or not can be easily checked via the points magnitude, i.e., $\sqrt{x^2 + y^2}$ compared to the radius of the circle.

[1] `http://www.studon.uni-erlangen.de/studon/goto.php?target=crs_1061841`

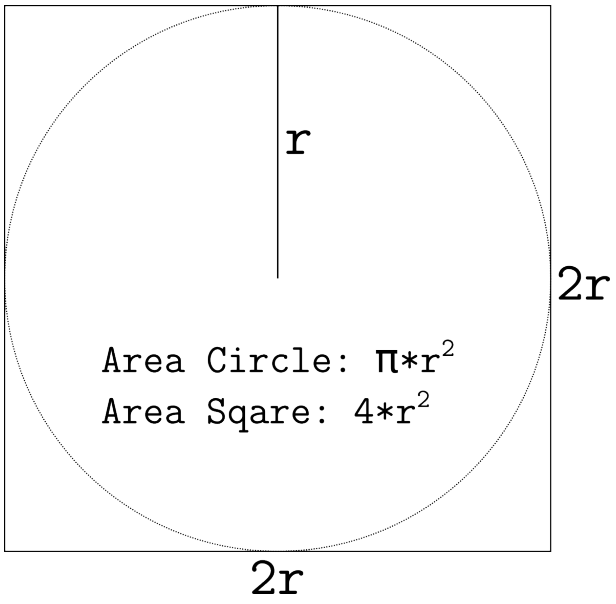

Figure 1: Visualization of the setup – circle inside square.

Random Numbers and Containers in C++ 11

Starting with the C++ 11 standard, the usage and generation of random numbers has changed fundamentally. Random numbers can be generated using a so called *random engine* and an appropriate *distribution*. Further details can be obtained here:

`http://www.cplusplus.com/reference/random/`

Containers of the standard library (std) are used to store collections of arbitrary objects (of the same type). The container objects provide functionalities (methods) for the management of the objects and further more take care of the memory management. For more information about containers see:

`http://www.cplusplus.com/reference/stl/`

Tasks

1 Simulation

Develop a Monte Carlo simulation for the estimation of Pi ($\approx 3.141592653589793238462\ldots$).

Use the Cartesian representation of complex numbers for your computation, i.e., reuse your class Complex from the previous assignment. You should create random complex numbers, i.e., with random real and imaginary parts. You then have to evaluate if it lies inside the circle or not. Use your *magnitude* function to check this. For the generation of the random numbers you have to use the *default_random_engine* along with an appropriate distribution. Use one vector to store the complex numbers and one to store the approximative values of Pi for each step of the approximation.

Define an appropriate stopping criterion for your simulation according to the convergence rate of

the results and/or a maximum number of random complex numbers.

2 Visualization

Your implementation has to provide a functionality to write the result of your simulation to a file; containing the approximative value of Pi along with the current iteration number (i.e., the number of random complex numbers created up to that point), for each approximation step. Visualize this output using gnuplot, excel,. . . (i.e., any other suitable tool). Make sure to include a plot for at least one run of your simulation in your submission; use pdf format.

3 Optimization

Optimize your solution by only considering one quarter of the area of the square and the circle. This way you can reduce the generation of random numbers to the domain of $re \in [0; 1]$ and $img \in [0; 1]$. Explain the impact and implications of this optimization on the parameters of you simulation. Also, include one plot generated from this optimized version in your submission.

More Remarks

Again, as in the previous project, we do not require you to use a particular coding style, but consider and try to stick to a consistent coding standard throughout the project. Also, *you have to document you project using doxygen, use the directory structure proposed for the previous project and make sure to provide a suitable makefile with sensible targets!*

The submission of the project will be via the StudOn course website. Please check regularly for updates and comments on the assignment and the format of the submission.