Assignment: Direct Simulation Monte Carlo and parallelization (lectures 5 & 6)

In this assignment you have to numerically obtain the value of π using the DSMC method (Task A) and then parallelize your code (Task B). This does not imply the usage of grid or processing of collisions, therefore, a very simple DSMC algorithm can be used to achieve the goal.

For this assignment, you should use the Python programming language, like in the assignments for the previous lectures. All knowledge of Python necessary for this task has been introduced previously (such as creation of functions and numpy arrays, plotting with Python, etc).

Task A. Calculate π using DSMC

Calculate π using the DSMC method. To achieve that, do the following:

1. Create a squared 2D domain (presented by minimum and maximum coordinates, i.e., create x_{min}, x_{max}, y_{min}, y_{max}).
2. Read the maximum number of particles to be launched, M, from the command line. Like that:

   ```
   python DSMC_pi.py 10000
   ```

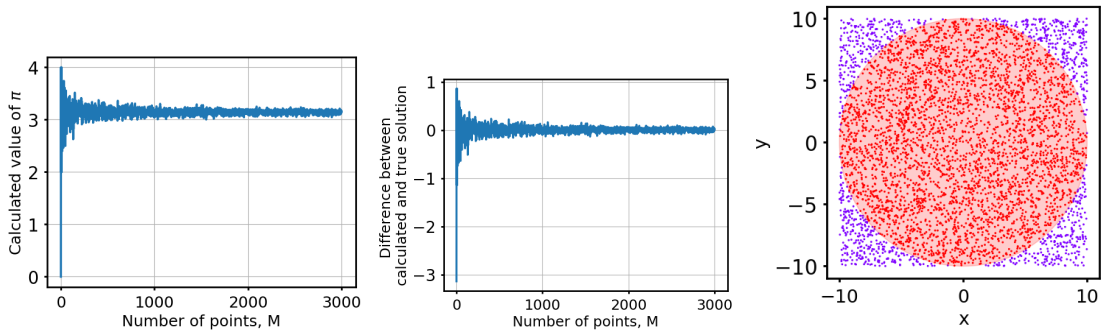
3. Write a function that launches n particles with random coordinates x_{rand} and y_{rand} and checks if a particle lies within the circle inside the square with the radius of the circle $(x_{max} - x_{min})/2$. Calculate the number of particles inside and outside of the circle for *all* $n \leq M$ (so, you will need to cycle M times inside your function).
4. Calculate π for all $n \leq M$ as a number of particles that fell inside the circle divided by total number of particles, times 4. Also, calculate the difference between your calculated π and the "real" π which you can get as numpy.pi.

Plot the results. Create two plots:

1. Plot the calculated value of π vs n.
2. Plot the difference between your calculated value of π and the value given by numpy.pi vs n.
3. Plot the domain, the circle inside, and the particles (make this plot only for a small M).

Run the program a few times to see that the plots look slightly different for a given M every time (because of the random principle of the DSMC method), but that the result always converges towards the real solution (correct value of π), if the number M is high enough.

Output should look similar to that:



Task B. Parallelize your code from Task A.

Parallelize your code for finding π using Python multiprocessing module (Pool or Queue, I recommend Pool since the task is simple, but Queue is also fine). Try to achieve a *net speedup* of the execution. This task is easy for parallelization: there are no data dependencies, work can be evenly divided between the processors, there is no need of communication or synchronization between processes.

Store only the value of π and the deviation $\pi_{calculated} - \pi_{machine\ precision}$.

1. Read from the command like the maximum number of the particles to calculate π (like in the previous assignment) + also the number of CPUs to run the program at.

2. Check, if the number of CPUs from the command line exceeds the number of available CPUs at the local machine, if yes, write an error message on the screen and set the number of CPUs to the maximum available at the computer.

3. Calculate π using your parallelized version of the DSMC function from the previous assignment and plot the results (π and precision vs the number of particles).

Using the time module, calculate the execution time for the same maximum amount of particles, M, for different cases:

1. Calculate the execution time at 1 CPU

2. Calculate the execution time at n CPUs (e.g., 2, 4...)

3. Calculate the net speedup/speeddown of the program

4. Print this data on the screen. Output should look something like that:

```
$ python3.5 DSMC_pi_parallelized.py 20000 4
Execution time at 1 CPU(s) equals 155.86495327949524 seconds
Execution time at 4 CPU(s) equals 55.031981229782104 seconds
Speedup 2.8322613468828655
```

```
$ python3.5 DSMC_pi_parallelized.py 100 2
Execution time at 1 CPU(s) equals 0.004492044448852539 seconds
Execution time at 2 CPU(s) equals 0.10972118377685547 seconds
Speedup 0.040940539412955994
```

You can see that the speedup can be achieved only for a sufficiently large number of particles.

You should write a very small and simple report presenting your results (like for previous assignments), just a few plots and a very short description, and send it to kristina.kislyakova@univie.ac.at. Please also send your code.