

## Assignment 1

Due: 20/11/2018, 11:00

In this assignment we implement a simple long term weather simulation. As dynamic model we use a transition matrix  $P \in \mathbb{R}^{n \times n}$  whose entry  $P_{ij}$  describes the influence of the  $j^{\text{th}}$  grid point on the  $i^{\text{th}}$  one. The grid points form a regular grid  $(\theta_i, \phi_i)$  in spherical coordinates with  $N_\theta$  and  $N_\phi$  points in the two coordinate directions, see Fig. 1. To obtain  $P$  the grid points are linearized row-wise.

The long term normalized weather (e.g. the temperature) on the grid points is given by the steady state distribution  $\pi \in \mathbb{R}^n$  of  $P$ . It is mathematically described by

$$\pi = \pi P, \quad (1)$$

i.e.  $\pi$  is a left eigenvector of  $P$  with eigenvalue 1 (which is the largest eigenvalue of  $P$ ). Our objective is to compute the steady state  $\pi$ .

To obtain  $\pi$  we will use power iteration, which is one of the simplest algorithms to compute an eigenvector. It iterates the following two steps

$$x'_{k+1} = x_k P \quad (2a)$$

$$x_{k+1} = \frac{x'_{k+1}}{\|x'_{k+1}\|} \quad (2b)$$

until either a given number of iterations has been performed or a suitable convergence criterion is met (for us the former suffices but you are free to think about a convergence criterion). The initial vector  $x_0$  at step  $k = 0$  is a random vector ( $(x_0)_i \propto \mathcal{U}([0, 1])$  suffices for us) and then it can be shown that the algorithm converges to the eigenvector associated with the largest eigenvalue as  $k \rightarrow \infty$ .

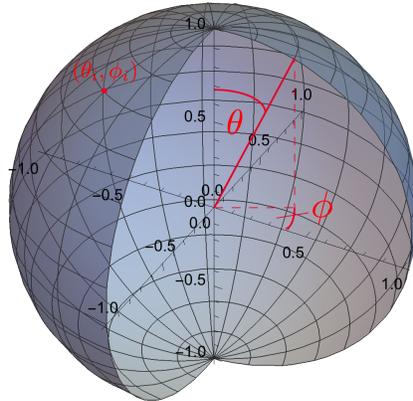


Figure 1: Regular grid in spherical  $(\theta, \phi)$  coordinates.

Your tasks are as follows:

- 1.) Download the [skeleton code](#) and generate the build system using `cmake`. The data files containing example transition matrices are also available for [download](#). These can be loaded using `readMat()` in the skeleton code.
- 2.) Implement a serial version of the power iteration algorithm to compute the stationary distribution  $\pi$  for the given transition matrices. (1 Point)
- 3.) Use `verifyStationary()` to verify that your solution is correct. Implement a suitable threshold for the return value of `verifyStationary()` so that a warning or error is generated when the accuracy goal is not met. (1 Point)
- 4.) Parallelize your serial implementation by computing the matrix-vector product in each iteration of power iteration in parallel. (3 Points)
- 5.) Parallelize your serial implementation using the task pool pattern and so that the threads do not have to be started again in each iteration. Use the barrier that is provided in the skeleton code to realize the required synchronization in the iterations. (3 Points)
- 6.) Generate a graph with the execution times of your serial and your two parallel implementations as a function of the grid resolution. Ensure that your measurements are accurate and meaningful to gauge the effectiveness of your implementations. (1 point)

*Bonus:* Implement the barrier without busy waiting. (2 points)

Please submit your implementation as well as the graph before the deadline to [gpu@isg.cs.ovgu.de](mailto:gpu@isg.cs.ovgu.de). Your code has to compile and run with the given CMake file.