Alumni Package for the simulation of the “Great Oxygen Event”

Abstract— **The following is explaining the program, written in c++, which is used for the simulations in my master thesis. The code contains a program, simulating a cellular automaton which is able to output the grid and the overall densities in text-files. For graphical virtualizations are taken third-party software. It can be done by open-source or Mathematica. This text explains how to run the program, print the grid, the overall densities and at least output the grid in a video.**

I. INTRODUCTION

In the following are presented two examples.simulations. One contains the model presented in the thesis on page 67 figure 3.33. This is a model which includes all – the author known– processes, including also oxygenic and anoxygenic photosynthesis for the Cyanobacteria, based only on rules without mathematical functions representing the HYPERBOLIC CONSERVATION EQUATIONS. The second code-example explained in this text is showing the simulation from page 72 figure 3.34 in the thesis. This simulation is realized with a minimum of rules only representing the Cyanobacteria and the Proteobacteria on the grid. All other resources are represented with HYPERBOLIC CONSERVATION EQUATIONS. The package should contain:

- a folder, called “example_1”, where the simulation without the HYPERBOLIC CONSERVATION EQUATIONS is included
- a folder, called “example_2”, where the simulation with the HYPERBOLIC CONSERVATION EQUATIONS is included
- a folder, called “files”, which includes
  - a bash-script, called “script”, to convert pdf-files to image-files
  - a text-based-file, called “kenorland.txt”, ready to import to the c++ code to represent a map of Kenorland based on the MOLLWEIDE-PROJEKTION
  - a text-based-file, called “world-map.txt”, ready to import to the c++ code to represent the world in its current constellation
  - a Mathematica -file, called “convert_jpeg.nb”, to convert a jpeg-image to a text-based-file ready to implement as an array into the c++ code
  - a Mathematica-file, called “density.nb”, to print the density slopes as pdf-file
  - a Mathematica-file, called “video.nb”, to print the grid in pdf-files

*This work was not supported by any organization

[1]H. Kwakernaak is with Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, 7500 AE Enschede, The Netherlands h.kwakernaak at papercept.net

[2]P. Misra is with the Department of Electrical Engineering, Wright State University, Dayton, OH 45435, USA p.misra at ieee.org

II. PROGRAMS

A. Structure of the programs

Each program contains 5 files. One file, named “main.cpp”, containing the main-program, two files, named “feld_ca.h” and “feld_ca.cpp”, representing the header -file (containing the declarations for the class feld_ca) and the cpp-file (containing the definition for the class), another two files , named “grid.h” and “grid.cpp”, representing also the header -file and the definition of the class.

feld_ca

This class contains all the needed variables for the grid represented by each cell. These can be, for example the iron concentration, the oxygen concentration in the water, the state of the bacteria and all other values for one cell in the grid. This class (“feld_ca”)is later included in the header file of the class grid. In the header file of the class grid (“grid.h”) is then declared an array, called “v2”, of the datatype “Feld_ca” defined by the class “Feld_ca”:

```
std::vector<std::vector<Feld_Ca>> v2 ;
```

Furthermore, the class “grid” contains all the needed functions, representing the rules for the GREAT OXYGEN EVENT. All these functions are normally called in “main.cpp” after making an object of the class “grid”.
The structure of the main-function is as follows: At the beginning are generated objects of the class “grid” twice. One object is containing the current state of the cellular automaton and the other object contains the state of the cellular automaton of the state $t + 1$. All the results of the functions, called in the main-function, are stored in the second object of the class grid and copied after every timestep to the first object of the class grid, to rerun the simulation again. This is done until the simulation stops. After creating the objects the functions

```
set_grid2(x, y,a);
set_grid(x, y);
```

are called. These functions are defining the size of the grid and reserving space in the Ram of the computer. Typically for Kenorland $x = 329$ and $y = 654$ are set as a size of the grid. The value “a” describes the number of the “Black Smokers” set randomly, at the beginning of the simulation into the water. The function

```
set_randomizer(false);
```

is defining how the random values are generated. Details should not be given here. Basic knowledge about “how random numbers are generated in c++” can be read in http://www.cplusplus.com/reference/cstdlib/rand/ to understand the setting of this function. Calling the function

```
set_random_grid_mit_karte("map.txt",a);
```

loads the map, saved in "map.txt", into the array of the vector "v2" defined in the class "grid". "a" is a value used in previous versions of the code and useless in this version of the code.
Before starting the loop there are called functions like

```
make_new_file_proteobacteria("ausgabe
                              /density");
```

which are creating files in the folder "ausgabe" or deleting them if they are not empty from previous runs. To this files is saved every timestep the state of the automaton.
After soem unimportant stuff (which is mentioned later), the loop is starting. Normally a run in the thesis contained 2,000 timesteps. At the beginning of each timestep, the densities are saved into the files and the automaton is printed in new file every 10th timestep. The densities are saved every timestep. A saving process of the automaton is realized with calling the function

```
print_grid_in_file_mathematica3(p,
        [](Feld_Ca & feld ){return
        feld.zustand_proteobacteria;});
```

which calles the template of the function "print_grid_in_file_mathematica3" with the lambda-function (available since c++ 11) "

(Feld_Ca & feld)return feld.zustand_proteobacteria;" here for the Proteobacteria.

System requirements:
All the simulations have been taken with a Linux Debian 9.3 with a kernel 4.9. Linux is not necessary but at least a Unix-based computer is needed to let the script running and FFmpeg is at the releasing date of this text not released in a Windows version. The source-code has to be downloaded and compiled for Windows. The settings of my system are:

- Debian 9.3 with Kernel 4.9
- gcc 7.3 (GNU Compiler Collection)
- FFmpeg 3.4.1
- Mathematica 8.0

There should be mentioned shortly, what the tools are doing.
gcc: This is a open-source based c++ compiler.
FFmpeg: This program is needed to illustrate the simulations in videos. It sticks the pictures, taken every 10th timestep, of the grid together and gives out a video as an avi-file. It has also other possibilities for video output formats.
Mathematica: Mathematica is a commercial product from the company Wolfram Research. It is not needed to use Mathematica but it makes it easier. It was also tested with gnuplot but Mathematica is easier and faster to handle. One of the tools is needed to plot the slopes of the densities and to print the grid. It is possible for both programs.

B. Example_1

As already mentioned, this example shows the result of figure 3.33 on page 67 for the choice between oxygenic or anoxygenic photosynthesis for the Cyanobacteria. The simulation results in the continuous slopes of figure 3.33.
The following discussion is based on the "main.cpp" located in the folder "example_1". Till line number 120 the all the main setting for the grid are made. Also, these lines are including settings for the Methane-concentration in the atmosphere of an integer value of 14,580,000.
The loop
After line number 120 the loop is starting which is here set to be repeated 2,000 times. Till line number 187 all the writing processes to the files are managed. It contains the call of 6 functions every timestep to write the overall densities and an if-case, valid every 10th timestep, including functions to print the grid.
After the saving processes, the diffusion is realized. This is happening till the line number 202. In line number 203 the state is saved to the object grid again. Then the photosynthesis for the Cyanobacteria is coming. As it is with the diffusion process there are two for-slopes running over all cells. In this slopes is checked if there is sitting a Cyanobacterium, continued if the oxygen is emitted into the ocean or atmosphere or checked if iron is present and done an anoxygenic photosynthesis with a possibility of 50%.
At line number 248 the reaction with iron and oxygen is starting. It is split, as mentioned in the thesis, in 3 cases.
The photosynthesis process for the Proteobacteria is starting at line number 280. After a successful photosynthesis process, a new neighbour is set randomly.
At line number 298 the dying out process for the Proteobacteria is realized if there is too much oxygen in the ocean present.
The next process starting on line number 314 is the reaction between Methane and oxygen in the atmosphere.
The penultimate step, starting on line number 339, is the outgassing of iron from the black smokers.
The last step is realized by calling the function

```
volcanism(o,a);
```

which realizes the boring billion. o is representing the atmospheric oxygen concentration and a the time step. This function decreases with an e-function the Methane concentration every time step.
After this, the loop is started again and begins with saving the calculated contents to files.

C. Example_2

The program discussed here contains the same structure as example_1. First, the grid is organized and all the files are created. Then the loop is starting, as already mentioned, starting with the saving processes for the density and followed by an if-case, true every 10th-time-step, to print the grid. Then the processes are organized like it is in example_1 and explained in the thesis in chapter 3.4 from page 67. It starts with the photosynthesis of the Cyanobacteria, continued by the reaction between iron and oxygen, the photosynthesis process for the Proteobacteria and the last reorganized step that the Proteobacteria are dying out if there is too much oxygen present. The next steps are the same as in example_1. The function of the Black Smokers are not existing anymore This is considered in the iron-oxygen reacting functions and of the remaining iron concentration is based the dying out process of the Proteobacteria. The reaction for the methane-oxygen reaction in the atmosphere and the vulcanism is the same and after this, the loop continues from the beginning.

III. Outputs

A. Running the program

Compiling now the program with gcc by running in a Unix-Terminal

```
g++ -Wall -o run main.cpp
```

gives out a file called "run" which is executable with

```
./run
```

also in a Unix-Terminal. The compiler also gives out warning messages with the explained command. By letting the program running files are saved into the folder "ausgabe". The files containing names with "density" are including in each case the density-slopes. The other ones are the pictures of the grid.

B. Printing the slopes

For this, there is executed the Mathematica file "density.nb". In the first line of the Mathematica notebook there has to be set the folder, where the folder "ausgabe" is located at the computer. Then the code saves a pdf-file with a legend including all slopes represented by a diagram. This can also be done by taking gnuplot.

C. Printing the Grid

For this, there is taken the Mathematica notebook "video.nb". At the first line, it has also to be set the location where the folder "ausgabe" is located. After this, there is saved for every 10th-time-step a pdf-file illustrating the grid, in a vector-based way. This is done for the Proteobacteria, Cyanobacteria, Ocean-Oxygen-Concentration and the Iron concentration. The pictures can now be viewed manually by opening every file.

D. Making a video

It should be shown how a video can be generated from the above-generated pdf-files showing an illustration in each case for the grid. For this, there is first needed the bash-script, called "script", which is converting the above-generated pdf-files to png-files. The script has to be copied in the folder "ausgabe". It can be executed with a Unix-Terminal by

```
sh script.
```

After this, in the folder "ausgabe" can be found for each pdf-file a png-image. These images can now be formed to a video with the open-source program FFmpeg. This can be done by

```
ffmpeg -r 5 -s 1920x1080 -start_number 0
      -i Proteobakterien_%.pdf.png
       proteobacteria.avi
```

which saves a avi-file for the Proteobacteria, calles "proteobacteria.avi".

E. Creating a map

Last but not least should be explained how to generate a map that is at the end ready to be imported to the c++ program as an array. This is also done with a Mathematica notebook in the attachment, called "convert_jpeg.nb". For best results, the jpeg which is read in has to contain black-white colours only. In the first line, the direction where the jpeg is located has to be defined. Then the Mathematica notebook blots an array. Each entry of the array is containing a vector. This vector explains the RGB-colours of each from the jpeg. It is then asked with an if-case if it contains a black or a white pixel, which is saved by a value of 0 or 1 to an array. This array is then saved as a csv-file. Removing the "," in this csv-file and replacing them with spaces gives a file which can be read in into the c++ program.

Appendix

- a folder "example_1" containing the first example
- a folder "example_2" containing the second example
- a folder "files" containing the Mathematica notebooks, a bash file and some maps

If there are some ongoing questions feel free to ask.