Victor-Ioan Macovei
Computational Physics MSc
Babeș-Bolyai University

July 9, 2024

HOMEWORK 1 — Warm-Up

Requirements and Setup

Resources

- **Development Environment**:
  - Visual Studio Code with C/C++ extension by Microsoft (required but can be replaced);
  - Python: Version 3.x. (required);
  - VS Code Extensions: Python, GitLens, GitHub Pull Requests and Issues (not required);
- **Compiler and Debugger**: GCC for C/C++ and GDB for debugging (required but can be replaced);
- **Version Control**: Git for cloning the repository (not required);
- **Python Dependencies**: Use of `pip` to install requirements from a `requirements.txt` file (required);
- **Continuous Integration**: GitHub Actions for Continuous Integration (not required).

Setup Guide

- **Cloning the Repository**: The repository was created on the local machine using Git. It is available at GitHub - ssmHW;
- **Setting Up the Development Environment**:
  - Visual Studio Code was downloaded and installed;
  - Required extensions, including "C/C++" by Microsoft and optionally Python, GitLens, GitHub Pull Requests, and Issues, were installed through VS Code's Extensions;
- **Installing Compiler and Debugger**: For Windows, MinGW, which includes GCC and GDB, was installed. The MinGW bin directory was added to the system's PATH environment variable;
- **Configuring the Development Environment**: `tasks.json` and `launch.json` were correctly configured, ensuring the `miDebuggerPath` in `launch.json` was updated to the correct path of GDB on the machine, if different from the default;
- **Building and Running the Program**:
  - The program was compiled utilizing the default build task defined in `tasks.json`;
  - For debugging, the Debug view was switched to, the appropriate configuration selected from the dropdown menu;
- **Running Python Programs**: Python programs were run by setting the working directory to `ssmHW` and executing `python week1_WarmUp/ex1_SingleDieThrowing/src/simulate_dice.py` in terminal;
- **Continuous Integration**:
  - GitHub Actions for Continuous Integration was utilized, with workflow configurations found in `.github/workflows`, automatically running specified workflows for Python and C programs upon commits and pull requests;
  - Finally, the project was pushed to GitHub.

1 Continuous Model

The continuous model simulates the behavior of rats in a continuous space, focusing on the relationship between the number of rats and the level of aggressivity. The goal is to study how the density of rats affects the overall aggressivity in the system.

General Approach

The approach involves simulating rats in a continuous space where each rat can detect others within a certain radius. The simulation iterates through multiple cycles, resetting rat positions, checking neighbors within a given radius, and updating their aggressivity states. The results are analyzed by varying the perception radius and recording the mean aggressivity.

Source Files Documentation

`main.cpp` - Main program to run continuous or discrete model simulation:

- **int main():**
  - Parameters: None
  - Description: Prompts the user to select either the continuous or discrete model and runs the corresponding simulation. Uses `ContinuousModel` or `DiscreteModel` based on user choice.

`Constants.h` - Defines constants and types for the simulation:

- **typedef double real;**
- **const real PROB_SPONTANEOUS = 0.01;**
  - Description: Probability of a calm rat becoming nervous spontaneously.
- **const int T_MAX = 100;**
  - Description: Maximum number of simulation cycles.
- **const real L = 1.0;**
  - Description: Side length of the area for the continuous model.
- **const real R_MIN = 0.0001;**
  - Description: Minimum perception radius.
- **const real R_MAX = 0.035;**
  - Description: Maximum perception radius.
- **const real R_STEP = 0.001;**
  - Description: Step size for varying the perception radius.

`ContinuousModel.h` **and** `ContinuousModel.cpp` - Implements the continuous model:

- **ContinuousModel(int numRats):**
  - Parameters: `int numRats`
  - Description: Constructor that initializes the continuous model with a specified number of rats. Sets default simulation parameters and reserves space for the rat population.
- **void simulate():**
  - Parameters: None

- Description: Runs the simulation by iterating through different perception radii and cycles, resetting rat positions, checking neighbors, and recording mean aggressivity. Outputs results to a file.

- **void reset():**
  - Parameters: None
  - Description: Resets the positions of all rats randomly within the area.
- **void checkNeighbors(real radius):**
  - Parameters: `real radius`
  - Description: Checks the neighbors of each rat within a specified radius and updates their aggressivity state based on the rules.
- **double meanAggressivity():**
  - Parameters: None
  - Output: `double`
  - Description: Calculates and returns the mean aggressivity of the rat population.

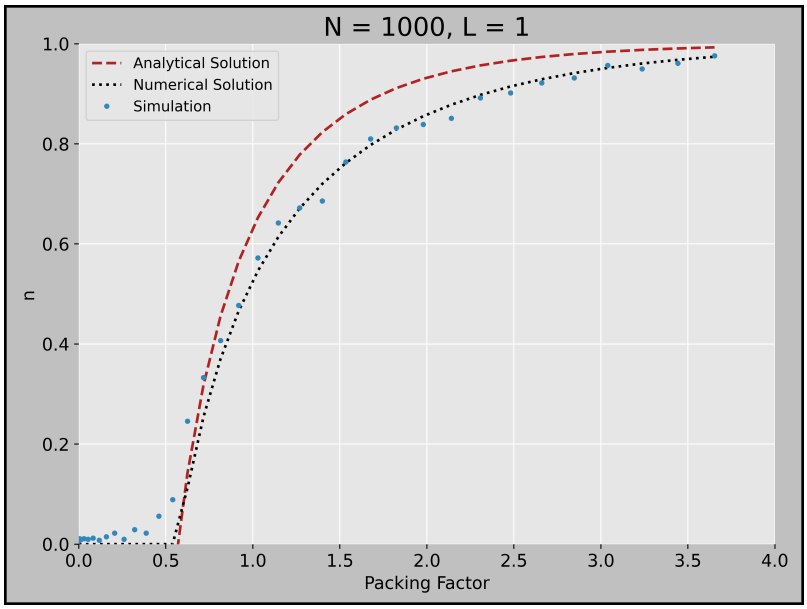

Figure 1: Nervous rat concentration as a function of the packing factor

2 Discrete Model

The discrete model simulates the behavior of rats on a discrete grid, focusing on the relationship between the number of rats and the level of aggressivity. The goal is to study how the density of rats affects the overall aggressivity in the system using a grid-based approach.

General Approach

The approach involves simulating rats on a grid where each cell can be empty, contain a calm rat, or contain an aggressive rat. The simulation iterates through multiple cycles, updating rat states based on their neighbors. The results are analyzed by varying the grid size and recording the mean aggressivity.

Source Files Documentation

`DiscreteModel.h` **and** `DiscreteModel.cpp` - Implements the discrete model:

- **DiscreteModel(int gridSize, int maxRats):**
  - Parameters: `int gridSize`, `int maxRats`
  - Description: Constructor that initializes the discrete model with a specified grid size and number of rats. Sets default simulation parameters and initializes the grid.
- **void simulate():**
  - Parameters: None
  - Description: Runs the simulation by iterating through different cycles, updating rat states based on their neighbors, and recording mean aggressivity. Outputs results to a file.
- **void reset():**
  - Parameters: None
  - Description: Resets the grid by randomly placing rats in empty cells.
- **void checkNeighbors():**
  - Parameters: None
  - Description: Checks the neighbors of each rat and updates its state based on the rules.
- **double meanAggressivity():**
  - Parameters: None
  - Output: `double`
  - Description: Calculates and returns the mean aggressivity of the rat population.

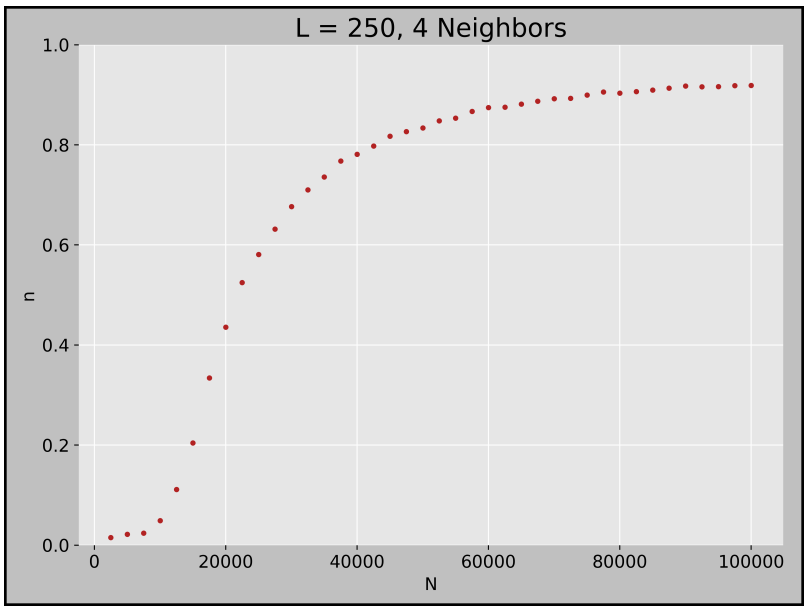

Figure 2: The nervous rat concentration as a function of the number of the rats.

3 Visualization

Visualization plays a crucial role in understanding the behavior and dynamics of the rat simulation models. This section involves plotting the results to provide insights into the aggressivity patterns and the phase transition as a function of rat density.

General Approach

The approach involves using Python scripts to read the simulation output files and plot the results. The plots include aggressivity vs. density curves, illustrating the phase transition, and other relevant visualizations to enhance the understanding of the simulation dynamics.

Source Files Documentation

`visual_analysis.py` - Visualizes the simulation results:

- **def plot_aggressivity_density(file_name):**
  - Parameters: `file_name`
  - Description: Reads the simulation output from the specified file and plots the aggressivity vs. density curve.
- **def plot_phase_transition(file_name):**
  - Parameters: `file_name`
  - Description: Reads the simulation output from the specified file and plots the phase transition curve, showing the critical density.
- **def plot_continuous():**
  - Parameters: None
  - Description: Plots the results of the continuous model, including the analytical and numerical solutions.
- **def plot_discrete():**
  - Parameters: None
  - Description: Plots the results of the discrete model, showing the relationship between the number of rats and the mean aggressivity.

Submitted by Victor-Ioan Macovei on July 9, 2024.