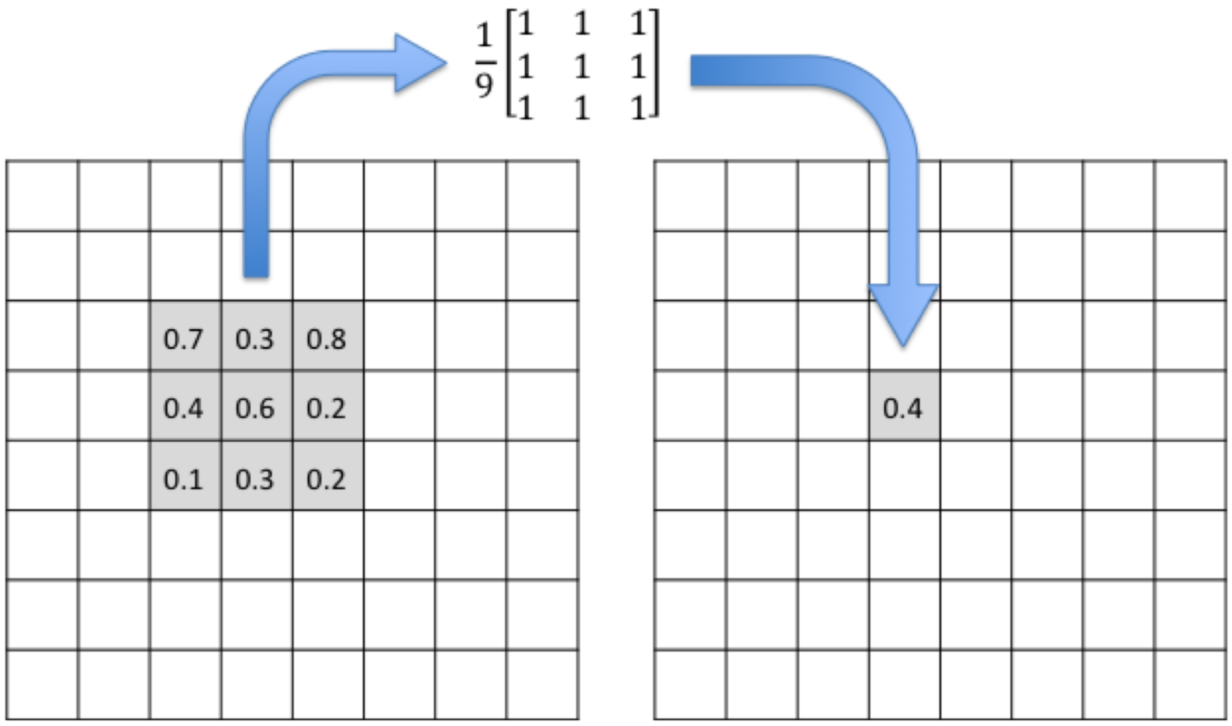
CME212 – Final Exam

March, 2018

Contents



$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$
0.7
0.3
0.8
0.4
0.6
0.2
0.1
0.3
0.2
0.4

Figure 1: Illustration of an implementation of a box-average diffusion model.

1 Problem Description

1.1 Belousov-Zhabotinsky Reaction

Your assignment is to model Belousov-Zhabotinsky diffusion-reaction system using cellular automaton approach, and demonstrate through simulation that simple chemical reactions can produce self-organizing behavior in homogeneous media. In a diffusion-reaction problems, concentrations of reactants change in time driven by reaction rates and diffusive transport. There is a number of chemical reaction models that fall into the class of Belousov-Zhabotinsky problems. We will use a simplified version defined by the following reactions

$$\mathrm{A} + \mathrm{B} \xrightarrow{\mathrm{k_A}} 2\,\mathrm{A}, \tag{1}$$

$$\mathrm{B} + \mathrm{C} \xrightarrow{\mathrm{k_B}} 2\,\mathrm{B}, \tag{2}$$

$$\mathrm{C} + \mathrm{A} \xrightarrow{\mathrm{k_C}} 2\,\mathrm{C}, \tag{3}$$

where A, B and C are (abstract) chemical compounds, and k_A, k_B and k_C are chemical reaction rate constants. We assume that chemical compounds form a thin layer on a surface of a substrate, and that their motion can be approximated as a diffusion on a 2-dimensional plane.[1] For simplicity, we will model this problem using a cellular automaton approach. This exam problem is based in part on a paper by Alasdair Turner; the last page contains prototype C-code you may find helpful.

1.2 Cellular Automaton

Cellular automaton is a collection of cells with certain properties and the rules describing how these properties change over discrete time steps. In our case, the cellular automaton is a 2-dimensional grid of identical cells. Chemical composition in each cell at grid position (i, j) is described by mass fractions y_A^{ij}, y_B^{ij} and y_C^{ij} of chemical compounds of A, B and C, respectively, and $i = 1, \dots, M$, $j = 1, \dots, N$. Due to mass conservation law, mass fractions have to add up to one:

$$y_A^{ij} + y_B^{ij} + y_C^{ij} = 1, \quad \forall\, i, j \tag{4}$$

For simplicity, we will assume that the substrate is rectangular, so the grid and the cells are rectangular, as well. We will also assume periodic boundary conditions for the cells.

The evolution rule consists of diffusion and reaction steps taken in succession. We will model diffusion using box average approach, i.e. we update value in each cell with the average of the values in that cell and its neighbors (see Figure 1).

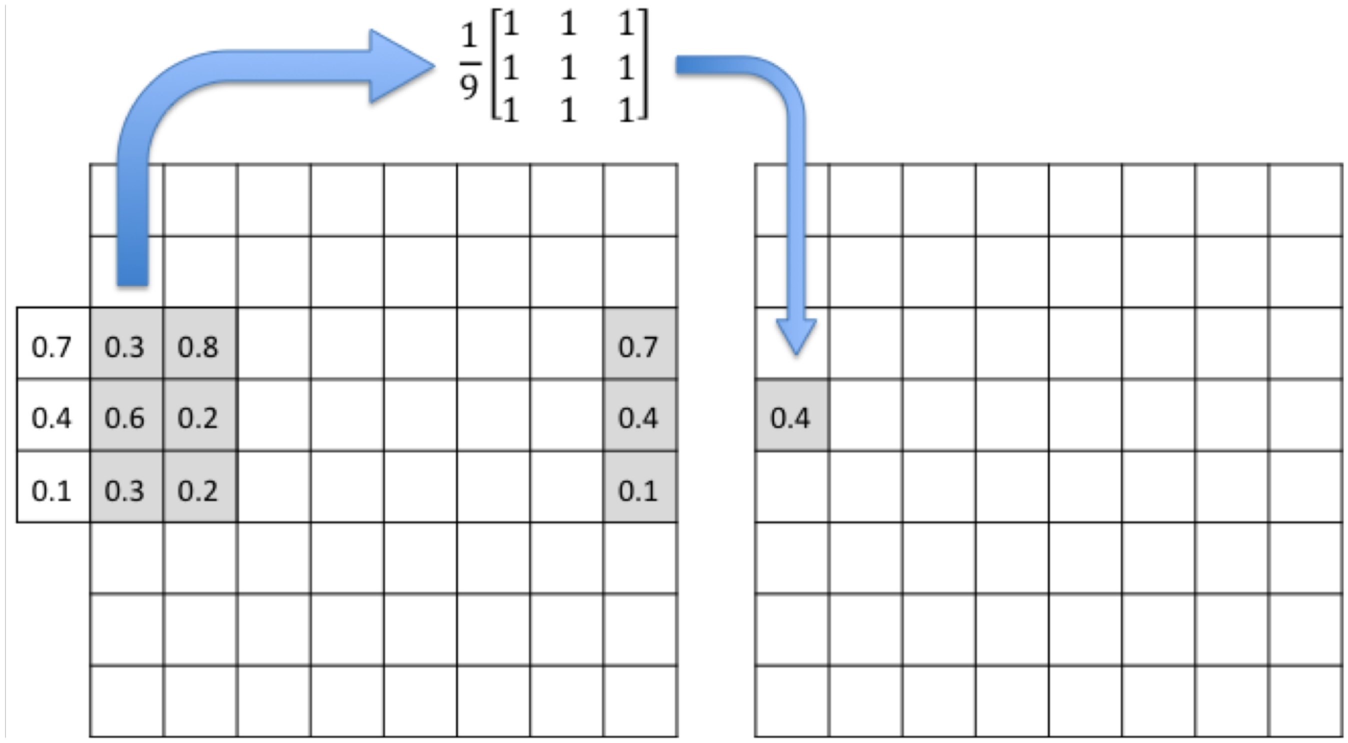
[1] Diffusion-reaction problems are typically considered continuous in space and time and are modeled by partial differential equations.

Mathematically, this can be represented like this:

$$y_\alpha^{ij} \leftarrow \frac{1}{(2m+1)(2n+1)} \sum_{k=-m}^{m} \sum_{l=-n}^{n} y_\alpha^{kl}, \qquad \alpha = A, B, C \tag{5}$$

The sizes of the cell neighborhood m and n used in the box average method represents the diffusion coefficients in horizontal and vertical directions, respectively. In the example in Figure 1, we set $m = n = 1$. **The diffusion is computed for each species separately**, and serves as the coupling mechanism between *cells*.

We use periodic boundary conditions in our model, therefore neighbors of the cell at a grid edge include cells at the opposite edge as shown in Figure 2.



$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$
0.7
0.3
0.8
0.7
0.4
0.6
0.2
0.4
0.1
0.3
0.2
0.1
0.4

Figure 2: Box-average diffusion model implementation with periodic boundary conditions.

Chemical reactions are computed and species mass fractions updated using discrete map

$$y_A^{ij} \leftarrow y_A^{ij} + y_A^{ij}(k_A y_B^{ij} - k_C y_C^{ij}), \tag{6}$$
$$y_B^{ij} \leftarrow y_B^{ij} + y_B^{ij}(k_B y_C^{ij} - k_A y_A^{ij}), \tag{7}$$
$$y_C^{ij} \leftarrow y_C^{ij} + y_C^{ij}(k_C y_A^{ij} - k_B y_C^{ij}), \tag{8}$$

where y_A^{ij}, y_B^{ij} and y_C^{ij} are mass fractions of A, B and C, respectively, in cell at the grid point (i, j). These equations follow directly from the chemical reactions (1)-(3). **Chemical reactions are computed for each cell separately**, and serve as the coupling mechanism between chemical *species*.

The discrete time steps may introduce numerical artifacts and produce nonphysical results. I.e. when mass fractions are near zero, the discrete time step may bump them down to nonphysical negative values, from which computations typically cannot recover. To ensure mass fractions have physical values, we set zero as the floor for each mass fraction value:

$$y_\alpha^{ij} \leftarrow \max(0, y_\alpha^{ij}), \quad \forall\, i, j, \alpha \tag{9}$$

The iteration (6)-(8) also may not preserve the mass conservation. To enforce mass conservation, the mass fractions may need to be renormalized after each reaction update

$$y_\alpha^{ij} \leftarrow \frac{y_\alpha^{ij}}{\sum_\beta y_\beta^{ij}}, \quad \forall\, i, j. \tag{10}$$

Here $\alpha, \beta \in \{A, B, C\}$, and $i = 1, \ldots, M$, $j = 1, \ldots, N$.

2 Assignment

2.1 Functionality Requirements (60 points)

Implement a cellular automaton class with the cell structure and the update rules as described in the previous section and demonstrate spiral pattern formation in the Belousov-Zhabotinsky system. A snapshot of a solution is shown in Figure 3. Your output may have different appearance, depending on the resolution and the color mapping you implement. ¡¡¡¡¡¡¡ HEAD

- (5 pts) Design a simple (template) data-structure to hold the chemical state of a cell.
- (10 pts) In `main`, initialize cells to random values, iterate the automaton, and visualize its evolution: e.g. Algorithm 1.
- (5 pts) Use 3×3 box average to model diffusion, and set chemical reaction constants to $k_A = k_B = k_C = 1$. A grid of size 100×100 cells should be sufficient to show pattern formation.
- (10 pts) Paint each cell red, green or blue with intensities proportional to chemical species mass fractions in that cell.[2]
- (10 pts each) Compute diffusion by species, and reactions by cell.
- (5 pts) Refresh the display after each automaton step. Class `Viewer` and example of its usage in file `main.cpp` are provided to help you visualize your results.
- (5 pts) The supplied `Makefile` contains settings to build your code with SFML library, which is used by the `Viewer` class. You will need to edit the `Makefile` to add instructions for building your object files. Your code must build without any warnings on **Ubuntu 16.04 virtual box** with following compile flags: `-std=c++11 -Wall -Wconversion -Wext`.

Algorithm 1 Diffusion-Reaction Computation

```
1: procedure
2:     Initialize all $y_\alpha^{ij}$ to random values in $[0, 1)$.
3:     $y_\alpha^{ij} \leftarrow y_\alpha^{ij} / \sum_\beta y_\beta^{ij}$
4:     while SFML window is open do
5:         Compute diffusion for each species and update mass fractions $y_\alpha^{ij}$.
6:         Compute reactions in each cell and update mass fractions $y_\alpha^{ij}$.
7:         $y_\alpha^{ij} \leftarrow \max(0, y_\alpha^{ij})$
8:         $y_\alpha^{ij} \leftarrow y_\alpha^{ij} / \sum_\beta y_\beta^{ij}$
9:         Update display.
```

2.2 Code Design (20 points)

You have freedom in choosing your design. You *are allowed* to make any changes to the `Viewer` class provided to you for this exam. A good code design allows you to make code changes or add new features with minimal disruptions to your code structure and interfaces. Here we provide sample guidelines for a good design of the B-Z model; each worth 2 pts.

- Cellular automaton model should be implemented as a C++ class.
- Cellular automaton class public methods should not depend on cell data memory layout.
- A user should be able to choose any rectangular grid of cells.
- A user should be able to set the size of the box used in the box average diffusion model.
- Interface to the automaton update rule should be clearly defined; enabling a future devloper to replace it with another.
- The diffusion and the reaction update should be implemented in separate methods, enabling independent modification.
- The interface to the reactions in the cell should be clearly defined; enabling future developers to replace with different reaction models easily.
- The number of reactants in the reaction model should not be hard wired to 3. The interface should support an arbitrary number of reactants with understanding that only mass fractions of three reactant can be visualized at the time.
- Data types should not be hard-wired. You should use class and function templates with data types as template parameters where appropriate.

[2]For example, intensity of red color represents mass fraction of A, green represents mass fraction of B, and blue mass fraction of C.

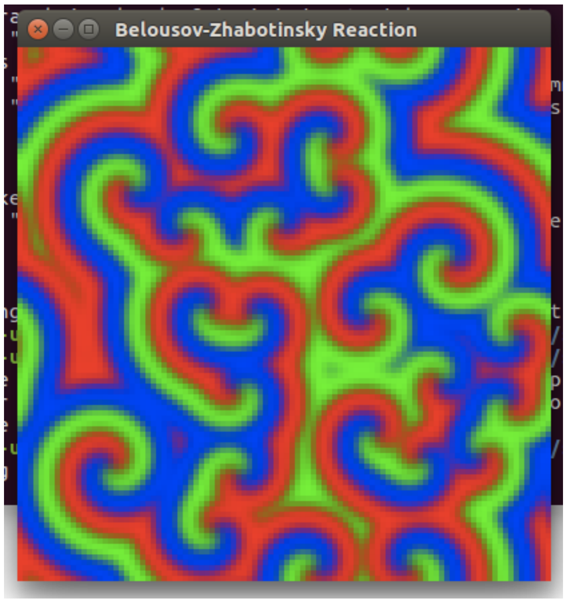


Figure 3: A snapshot of a solution showing characteristic Belousov-Zhabotinsky spiral patterns.

Again, the bullet points above are just guidelines. **All the design choices are yours to make**. Improving code design is usually an open ended problem, so you should **balance your design decisions with the time you have to complete this exam** and the price your customer is willing to pay for the improved design (20 points in this case).[3] Do justify your design decision in the documentation.

2.3 Style (5 points)

Your implementation style should be consistent throughout your code and chosen so to make your code easy to read.

2.4 Documentation (15 points)

You should provide a `README` file with a brief description of your code and justification of design decisions you made.

- (3 pts) In code comments, specify pre/post conditions for key methods.
- (3 pts) Identify representation invariants.
- (3 pts) Use of doxygen style comments.
- (2 pts) Decisions surrounding memory management and RAII.
- (2 pts) Discuss how move semantics are (not) used in your class.
- (2 pts) Mention how concepts could be used to improve your templated class.

3 Exam submission

You should create directory `final` in your GitHub repository and push your complete solution there. Your submission should include:

- All source and make files you need to build your solution.
- A `README` file with code documentation and build instructions.

Your solution **must build and run on Ubuntu 16.04 virtual box**. You must not include any binary files with your submission. Also, please remove any remaining `DummyClass` code from your submission.

[3] You'll notice there are only 9 bulleted items, i.e. we give you concrete ideas for how to obtain 18 points. The remaining two points may be awarded for exemplary-submissions; we cap this number at two so that we discourage students from spending *too much* time working on the final exam. Please write one sentence describing why the customer should pay these extra two points, in the case of your particular program.

4 A Note on Testing Recommendations and Partial Credit

To test parts of your code, you can run standalone diffusion and reaction modules each connected to the viewer separately. We emphasize that this is *not required* for full-credit, but may be helpful in debugging or in earning partial credit in the event you are not able to get a working solution.

If you start from a grid of randomly initialized cells, the correct diffusion algorithm will produce a grid of cells uniformly colored dark gray (i.e. $y_A \approx 0.33$, $y_B \approx 0.33$, $y_C \approx 0.33$) after a few iterations. To test the reaction module, first initialize grid uniformly (e.g. set $y_A = 0.3$, $y_B = 0.4$, and $y_C = 0.3$, in all cells), and then iterate reactions. If reactions are correctly implemented, the cells will change their color periodically.

Last Ditch Attempt at Partial Credit In an unlikely case you cannot produce pattern formation as shown in fig 3, submit (at least one) unit tests for each of diffusion and reaction modules to get partial credit, explaining the purpose of each test. Again, we emphasize that unit tests are not required to earn full credit on a working program, but that they can be used to earn partial credit on an incomplete program. You have to be able to visualize your results to get the partial credit for either module.