# Carleton University

**Department of Systems and Computer Engineering** **SYSC 5104 – Methodologies for Discrete Event Modelling and Simulation**

## Assignment 2

**Student Name:** William Wang **Student ID:** 101191168

GitHub repository:  
🔗 <https://github.com/RTGTX7/gray-scott-cellular>

### 1. Introduction

For this assignment, I have chosen to simulate the model presented in John E. Pearson’s paper *“Complex Patterns in a Simple System”* (Science, 1993), which investigates reaction-diffusion dynamics through the Gray-Scott model. This system simulates complex spatial patterns such as spots, stripes, and waves emerging from simple chemical interactions. The model is well-suited for implementation in the Cell-DEVS formalism due to its grid-based and time-evolving nature.

### 2. Overview of the Chosen Model

The Gray-Scott model at the core of Pearson’s work describes two reacting and diffusing chemicals, **u** and **v**, governed by the following partial differential equations:

**Reaction Equations:**

These equations contain both diffusion and reaction terms:

The diffusion terms, and ( ), describe the spatial spread of chemical concentrations. In the implementation, the Laplacian is approximated by summing the values of neighboring cells and subtracting four times the center value, following a fourth-order finite difference scheme.

The reaction term ( ) drives the nonlinear interaction between the two chemicals, where u is consumed and v is produced. This nonlinearity is key to generating complex spatial patterns.

The feed and decay terms control how much of each chemical is injected or removed. The expression f(1 - u) adds u into the system, while ( ) removes v from it.

Adjusting the parameters, one can produce various stable and dynamic patterns, such as spots, stripes, and waves.

### 3. Representation as a Cell-DEVS Model

The Cell-DEVS implementation of the Gray-Scott model consists of the following:

* **Cells:** Each cell holds state variables uu**u** and vv**v**, representing chemical concentrations. The state is updated based on both local reactions and diffusion from neighbors.
* **Neighborhood:** A Moore neighborhood (8 adjacent cells plus itself) is used for spatial interaction.
* **Transition Rules:** The update rules use the reaction-diffusion equations to evolve concentrations over time.
* **Coupling:** Diffusion across neighboring cells is modeled via coupling, while reaction occurs locally within each cell.

The grid uses periodic boundary conditions to ensure that the edges wrap around, allowing interaction across boundaries and forming a continuous surface for pattern propagation.

### Formal Specifications

#### Atomic Cell-DEVS

**GrayScottCell** (Atomic Cell-DEVS)

1. **X**: Set of external input events

* X =, where represents the concentrations from a neighboring cell.

1. **Y**: Set of external output events

* Each cell sends its updated concentration values to its neighbors. Hence,

1. **S**: Set of internal cell states

* In the grayScottState class, each cell holds:
  + Two real-valued concentrations:
  + A derived value v\_ratio, representing the ratio
* Simplified formally as:
* By default,

1. **N**: Neighborhood definition

* On a 2D discrete grid, a cell at position uses a Moore neighborhood of radius 1 (3×3 block including itself and 8 neighbors). Formally:

1. **τ**: Local computation function

* Defined in grayScott::localComputation(...), this function performs the core update step per time unit.
  + **Parameters** (from code, constants or configurable via JSON):
  + (Feed, Kill, and Reaction rates)
  + (Diffusion coefficients for chemicals A and B)
  + (Time step size)
  + **Discrete Laplacian calculation** A weighted 3×3 kernel is used to approximate the Laplacian:
  + **Reaction-diffusion equations** Let and represent the concentrations of chemicals A and B:
  + **Update step using Euler integration**:
  + **Additional computation**: The code also calculates with a fallback value (e.g., 0.5) to avoid division by zero.
* Therefore, the update function can be expressed as:

1. **δint**: Internal transition function

* In Cadmium’s GridCell class, this is tightly linked with localComputation(). When the internal timer expires (according to ), the state is updated using .

1. **δext**: External transition function

* If inputs from neighbors arrive earlier than scheduled, they are buffered until the next internal transition. Handled automatically by the GridCell framework.

1. **λ**: Output function

* After each internal transition, the cell outputs its updated state to its neighbors. Formally, the current state is the output.

1. **d**: Time advance function In the code, outputDelay(...) returns a fixed value of 1.0, meaning each cell triggers an internal event every 1 simulation time unit. Therefore,

#### Generic Coupled **Gray-Scott** (2D) Cell-DEVS

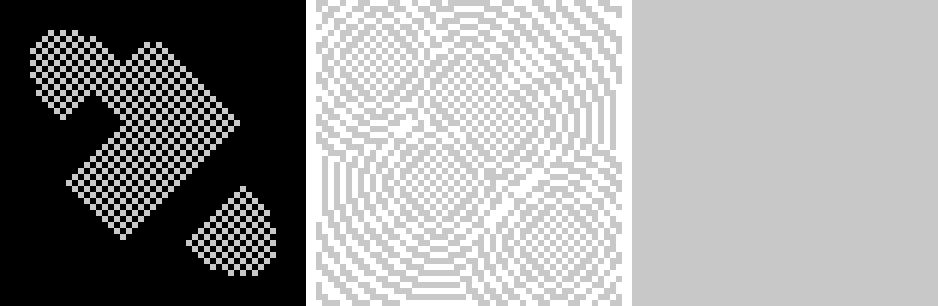
* Ylist = ∅ (No cell concentration export is specified in the configuration.)
* Xlist = ∅ (There is no external input to inject u,v concentrations into any cells.)
* I = (Trivial input interface, as both Xlist and Ylist are empty — no external ports are defined.)
  + For each , is undefined because .
  + There is no external coupling, so is unused.
* **X =**  **Y =**  (Each port’s message is the Gray-Scott concentration vector )
* (Represents the number of rows and columns of the grid, )
* (Total number of cells)
* N = ∧ , for all } (This represents the Moore neighborhood)
* $ C = { C\_{ij} i [0,m),, j [0,n) }$, where

is a Gray-Scott atomic cell, where:

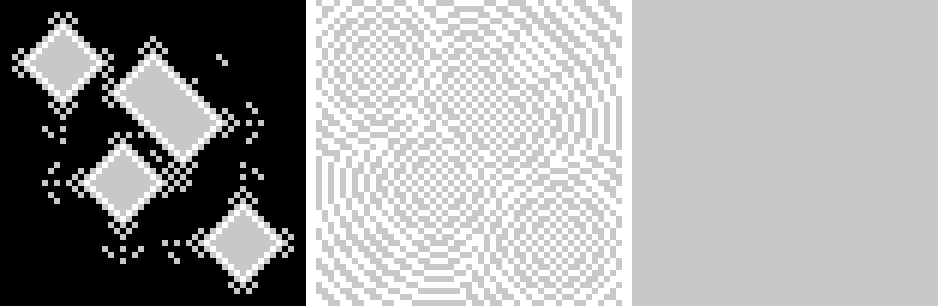
* The state
* The local update rule is defined in grayScott::localComputation()
* **B =**
* **Z:** , defined by:
  + , with , where .
* **select =** This is the tie-breaking function, which can be set to a default order (e.g., row-major order).



double f = 0.35; double k = 0.6497; double r = 1.0;

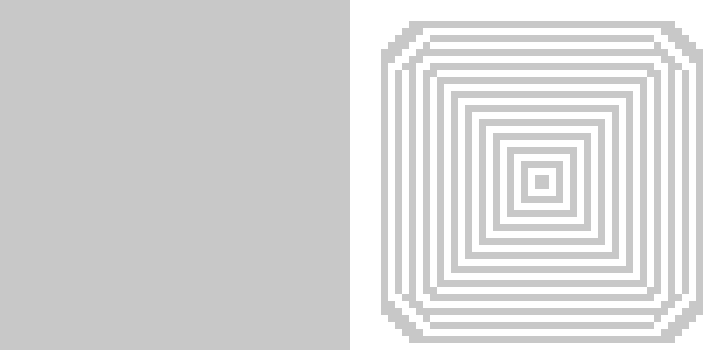
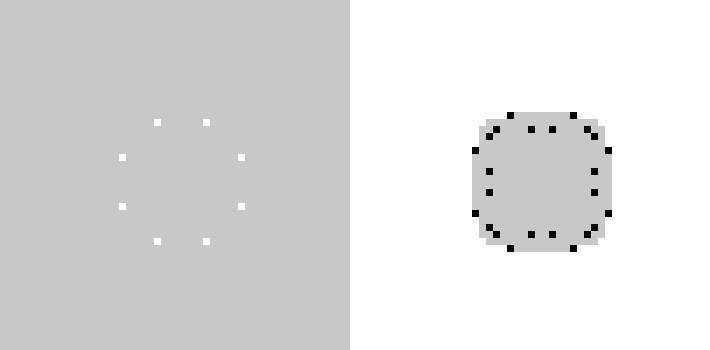
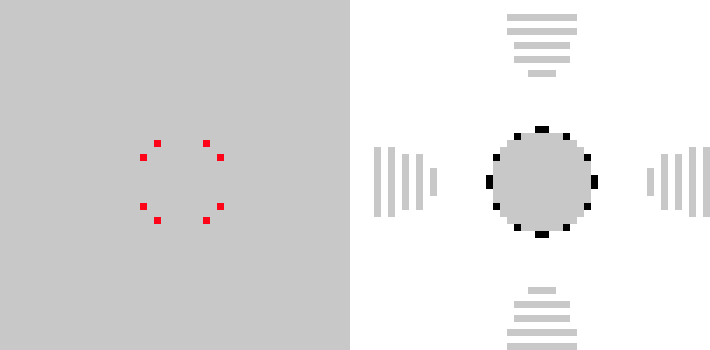
**Figure 1** uses parameters f = 0.35, k = 0.6497, and r = 1.0. The pattern shows a strong emergence of distinct B clusters in the left panel, while the A field (center) develops soft gradients and fluid transitions. The right panel appears mostly uniform, suggesting stabilization in that field. This demonstrates how a moderate feed rate and kill rate can produce structured yet stable behaviors. 

double f = 0.34; double k = 0.95; double r = 1.0;

​**Figure 2**​, with f = 0.34 and a higher kill rate k = 0.95, results in more compact and isolated B regions. The center panel reveals symmetric, wave-like gradients in A, while the right panel again appears uniform, indicating minimal change. This scenario shows how increased inhibition (higher k) dampens pattern spread and leads to localized spots. 

double f = 0.38; double k = 0.82; double r = 1.0;

​**Figure 3**​, where f = 0.38 and k = 0.82, highlights an early or intermediate stage of pattern formation. The B field produces sharp diamond-like structures, with peripheral noise possibly indicating ongoing development. The A field displays rhythmic gradients radiating outward, and the right panel remains gray and balanced. These settings generate highly geometric and symmetric distributions, illustrating the system’s sensitivity to small changes in parameters.

Some Other Results 

### Limitations

In contrast to other implementations of the Gray-Scott model, which typically simulate over 2000 or more time steps to observe full pattern emergence, this simulation required a compromise due to output constraints. High-resolution grids and frequent updates produced an overwhelming number of CSV log entries, making it difficult to control file size and numerical precision.

To address this, the simulation time step was increased by setting constexpr double dt = 2.0, effectively accelerating the reaction-diffusion process and reducing the number of iterations needed to reach visible patterns. While this adjustment speeds up pattern development, it also introduces trade-offs: numerical stability decreases, the simulation becomes more sensitive to minor fluctuations, and the resulting dynamics may deviate from the fine-grained behavior observed in longer, smaller-step simulations reported in the literature.

### Conclusion

This simulation successfully demonstrates the emergence of complex spatial patterns in the Gray-Scott model using a grid-based cellular automaton approach.

### Reference

* **John E. Pearson** (1993). ​*Complex Patterns in a Simple System*​. ​**Science**​, New Series, ​**Vol. 261**​, No. 5118, pp. 189–192. <https://www.researchgate.net/publication/6011915_Complex_Patterns_in_a_Simple_System>
* **Gray-Scott Model Overview** – Biological Modeling Project. <https://biologicalmodeling.org/prologue/gray-scott>