Simulation of Bi-molecular Reaction with Diffusion

Xinyu Li

xl3665@nyu.edu

October 28, 2022

1 Introduction

This project studies the following reversible bi-molecular reaction

$$\text{A+B} \underset{k_2}{\overset{k_1}{\rightleftharpoons}} \text{AB}$$

where k_1 is the reaction rate coefficient of the binding reaction, k_2 is the reaction rate coefficient of the unbinding reaction.

The reaction takes place on the two-dimensional surface of a torus formed by a 1 meter by 1 meter square. Figure 1 demonstrate how to "glue" the boundaries of a square to form this torus. This allows us to ignore the boundary of the square. For example, when a diffusing particle X hits

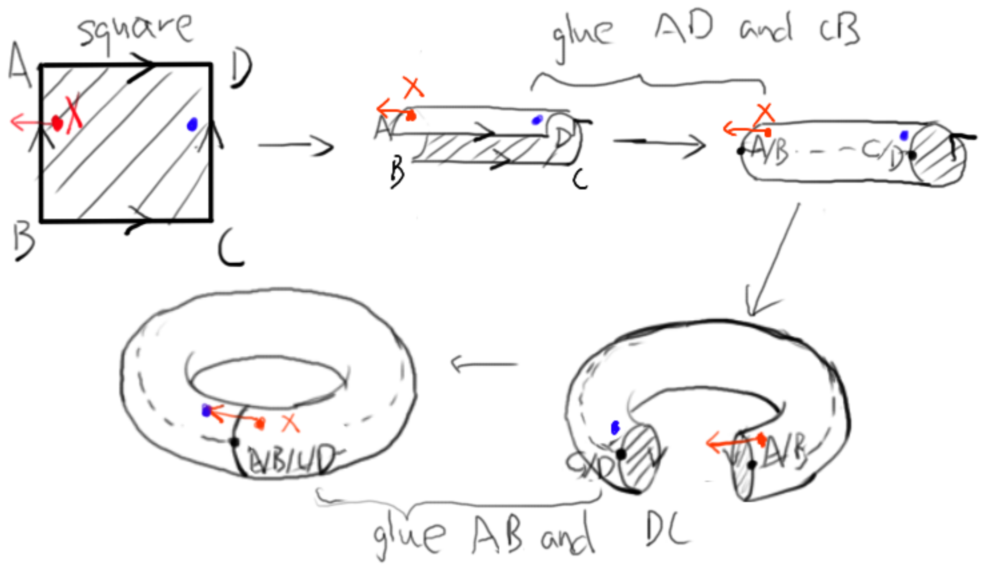

Figure 1: This figure shows how to turn a square into a torus

the boundary AB in figure 1, it will move to the opposite side CD because these two boundaries are glued together. Initially, there will be a given amount of A and B at different location on the torus (i.e. A and B are not well-mixed). No additional A, B or AB will be added or removed during the reaction.

My goal is to study the temporal and spatial evolution of the concentration of A, B and AB. Theoretically, I will state and explain the system of differential equations that govern the reaction. Numerically, I will use Euler method to discretize and solve the differential equations. Finally, I will tune the coefficients and initial conditions to see if there is any interesting phenomena in the simulation.

2 Equations

Let $v = (x, y) \in \mathbb{T}^2$ be the spacial location on the torus where $\mathbb{T}^2 = \mathbb{R}^2/([0, 1) \times [0, 1))$ is the torus wrapped from a $1m * 1m$ square. The unit of x and y is meter m. Let $t \in [0, \infty)$ be the time (s). Let $[X]$ denote the concentration (m^{-2}) of particle X. $[X] = [X](v, t) = [X](x, y, t)$ is a nonnegative function on $\mathbb{T}^2 \times [0, \infty)$. Let D_X be the diffusion coefficient $(m^2 \cdot s^{-1})$. Let $\triangle u(v, t) = \frac{\partial^2 u}{\partial x^2}(v, t) + \frac{\partial^2 u}{\partial y^2}(v, t)$ be the Laplacian of u evaluated at (x, t). The reaction is governed by the following system of differential equations [Fife(1979)]

$$\begin{cases} \frac{\partial [A]}{\partial t}(v,t) = -k_1[A](v,t)[B](v,t) + k_2[AB](v,t) + D_A \triangle A(v,t) \\ \\ \frac{\partial [B]}{\partial t}(v,t) = -k_1[A](v,t)[B](v,t) + k_2[AB](v,t) + D_B \triangle B(v,t) \\ \\ \frac{\partial [AB]}{\partial t}(v,t) = k_1[A](v,t)[B](v,t) - k_2[AB](v,t) + D_{AB} \triangle AB(v,t) \end{cases} \tag{1}$$

These equations essentially state that the rate of change of concentration is determined by the binding/unbinding reaction and diffusion. Take A as an example. The rate of change $\partial[A]/\partial t$ is minus binding rate $(-k_1[A](v,t)[B](v,t))$ plus unbinding rate $(k_2[AB](v,t))$ plus diffusion rate $(D_A \triangle A(v,t))$. Compared to typical diffusion equation $\partial[A]/\partial t(v,t) = D_A \triangle A(v,t)$, there is an extra term on the right that represents the contribution of chemical reaction.

Theoretically, since we do not add or remove any substance during the reaction, the total amount of each particle $[A^*] = [A] + [AB]$ and $[B^*] = [B] + [AB]$ should be conserved This is to say that $\int_{\mathbb{T}^2}[A^*](v,t)\,dv = C$, which can be simplified as follow

$$\begin{aligned} \int_{\mathbb{T}^2}[A^*](v,t)\,dv &= C_A \\ \frac{\partial}{\partial t}\int_{\mathbb{T}^2}[A^*](v,t)\,dv &= 0 \\ \int_{\mathbb{T}^2}\frac{\partial}{\partial t}[A^*](v,t)\,dv &= 0 \end{aligned}$$

Our equation are in line with this property. This can be proven by divergence theorem on manifold and the fact that the boundary of a torus is $\emptyset$.

$$\begin{aligned} \int_{\mathbb{T}^2}\frac{\partial}{\partial t}[A^*](v,t)\,dv &= \int_{\mathbb{T}^2}\frac{\partial}{\partial t}[A](v,t) + \frac{\partial}{\partial t}[AB](v,t)\,dv \\ &= \int_{\mathbb{T}^2} -k_1[A](v,t)[B](v,t) + k_2[AB](v,t) + D_A \triangle A(v,t) \\ &\quad + k_1[A](v,t)[B](v,t) - k_2[AB](v,t) + D_{AB} \triangle AB(v,t)\,dv \\ &= \int_{\mathbb{T}^2} D_A \triangle [A](v,t)\,dv + \int_{\mathbb{T}^2} D_{AB} \triangle [AB](v,t)\,dv \\ &= \int_{\mathbb{T}^2} D_A \nabla \cdot (\nabla [A](v,t))\,dv + \int_{\mathbb{T}^2} D_{AB} \nabla \cdot (\nabla [AB])(v,t)\,dv \\ &= D_A \int_{\partial \mathbb{T}^2} \nabla [A](v,t)\cdot dn + D_{AB}\int_{\partial \mathbb{T}^2} \nabla [AB](v,t)\cdot dn \\ &= D_A \int_{\emptyset} \nabla [A](v,t)\cdot dn + D_{AB}\int_{\emptyset} \nabla [AB](v,t)\cdot dn \\ &= 0 \end{aligned}$$

3 Numerical Method

We will apply Euler's method to find out a numerical solution. We cut the square into a $(100, 100)$ grid of spatial step size h meter. i.e. we cut 1m*1m square into 10000 hm*hm small squares.

Let the time step be δts, the finite difference version of the system of differential equations 1 is

$$\begin{cases} [A](v,t+\Delta t) = [A](v,t) + (-k_1[A](v,t)[B](v,t) + k_2[AB](v,t) + D_A \tilde{\triangle} A(v,t))\Delta t \\ \\ [B](v,t+\Delta t) = [B](v,t) + (-k_1[A](v,t)[B](v,t) + k_2[AB](v,t) + D_B \tilde{\triangle} B(v,t))\Delta t \\ \\ [AB](v,t+\Delta t) = [AB](v,t) + (k_1[A](v,t)[B](v,t) - k_2[AB](v,t) + D_{AB} \tilde{\triangle} AB(v,t))\Delta t \end{cases} \tag{2}$$

where $\tilde{\triangle} u$ is the discretized Laplacian of u defined as follows

$$\tilde{\triangle} u(x,y,t) := \frac{u(x+h,y,t)+u(x-h,y,t)+u(x,y+h,t)+u(x,y-h,t)-4u(x,y,t)}{h^2}$$

Using Taylor series, we can show $\tilde{\triangle} u$ approximates the real Laplacian $\triangle u$ with an error of order $O(h^2)$. For a function $f(x)$ smooth enough, we have

$$\begin{aligned} f(x+h)-f(x) &= f'(x)h + \frac{1}{2}f''(x)h^2 + \frac{1}{6}f^{(3)}(x)h^3 + \frac{1}{24}f^{(4)}(x)h^4 + O(h^5) \\ f(x-h)-f(x) &= -f'(x)h + \frac{1}{2}f''(x)h^2 - \frac{1}{6}f^{(3)}(x)h^3 + \frac{1}{24}f^{(4)}(x)h^4 + O(h^5) \\ f(x+h)+f(x-h)-2f(x) &= f''(x)h^2 + O(h^4) \\ f''(x) &= \frac{f(x+h)+f(x-h)-2f(x)}{h^2} + O(h^2) \end{aligned}$$

Thus, we have

$$\begin{aligned} \triangle u(x,y,t) &= u_{xx}(x,y,t) + u_{yy}(x,y,t) \\ &= \frac{u(x+h,y,t)+u(x-h,y,t)+u(x,y+h,t)+u(x,y-h,t)-4u(x,y,t)}{h^2} + O(h^2) \\ &= \tilde{\triangle} u(x,y,t) + O(h^2) \end{aligned}$$

To ensure the numerical stability of the simulation, a necessary condition is that the quantity $\mu = \frac{D\Delta t}{h^2}$ should be small enough [Tornber(2011)]. This is a dimensionless quantity which in essence means the number of h*h squares a molecule can move away in one time step Δt. For example, $\mu = 2$ means a particle X can move 2 h*h squares away from where it was in Δt time. A μ small enough is necessary for a numerically stable simulation.

4 Validation

As mentioned in Section 2, the total amount of the quantities $[A^*]$ and $[B^*]$ over the torus should be conserved. On the continuous torus, it means the integrals of $[A^*]$ and $[B*]$ over the torus are constants with respect to time. The discrete counterpart of this property is

$$\sum_{(x_i,y_j)} [A^*](x_i,y_j,t) = C_A$$
$$\sum_{(x_i,y_j)} [B^*](x_i,y_j,t) = C_B$$

where (x_i, y_j) are points on the grid.

By tuning the time step Δt and grid gap h, we can verify that these two quantities are conserved when the simulation is numerically stable. For simplicity, I will only show one example. In the next section, In section 5, I will empirically show that if the parameters satisfy that $D_X dt/h^2 \leq 0.25$, the simulation will be numerically stable and these two quantities will be conserved.

Figure 2 visually shows that these two quantities are conserved when $D_A = D_B = D_{AB} = 0.0002m^2/s$, $k1 = k2 = 0.23$, $h = 0.005m$, $dt = 0.01s$. The experiment lasts for $400s$ and the reaction reaches an equilibrium. A and B are placed on two concentric, adjacent rings on the square.

To ensure that the experiment does reaches equilibrium, I calculate time evolution of the standard deviation of all species over the torus. The result is displayed in figure 3. Note that when reaching equilibrium, the standard error should be very small since the species are almost well-mixed.

It is possible that $[A^*]$ and $[B^*]$ have great fluctuations during the reaction and such fluctuations are "smoothed" by the long time period and equilibrium. To ensure this is not the case, I calculate

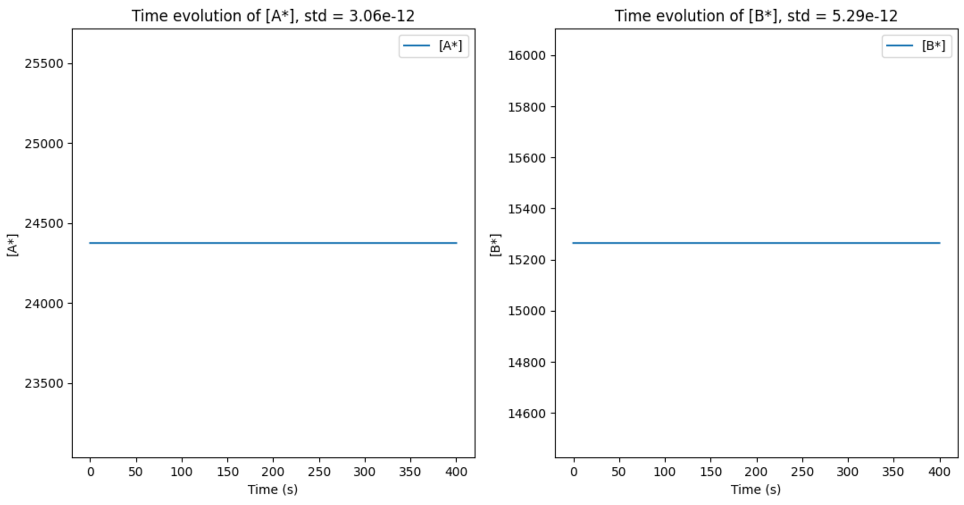

Figure 2: The time evolution of $[A^*]$ and $[B^*]$.

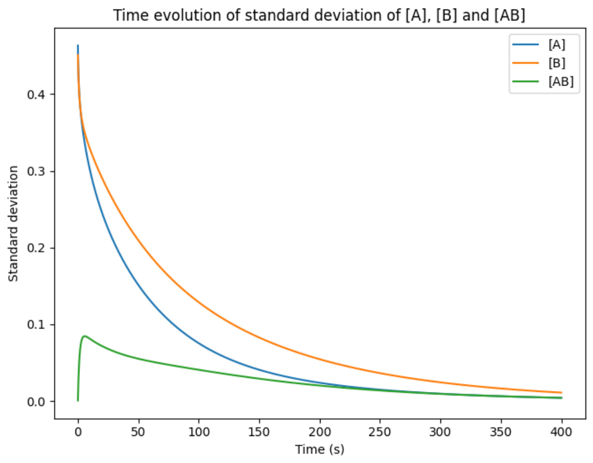

Figure 3: The time evolution of standard deviation of all species over the torus.

the standard errors of different time intervals of length $80s$ during the entire $400s$. The result is in table 1. We can see that the standard deviation over every time interval is relatively small, which further verifies that the quantities $[A^*]$ and $[B^*]$ are conserved over time.

t_0	t_1	$std([A^*])$	$std([B^*])$
0.00	80.00	3.04e-12	3.32e-12
80.00	160.00	3.06e-12	3.32e-12
160.00	240.00	3.08e-12	3.69e-12
240.00	320.00	3.04e-12	3.77e-12
320.00	400.00	3.07e-12	3.85e-12

Table 1: Standard deviation of $[A^*]$ and $[B^*]$ over different time invertals $[t_0, t_1]$ of duration 80s

5 Results and Discussion

I have conducted two experiments. The first experiment is about numerical stability of the simulation. The second one is about pattern formation under different initial conditions.

Experiment 1: Numerical stability The first experiment empirically shows that a necessary condition for numerical stability is that the dimensionless quantity $\mu := D_X \Delta t / h^2$ should be no larger than 0.25. This is the CFL (Courant-Friedrichs-Lewy) condition for this system of differential equations and this bound 0.25 is the so-called Courant number [Tornber(2011)]. Intuitively, μ represents the number of $h * h$ small squares on the grid a particle can go through in one time step Δt.

I find that a typical sign of numerical instability is the appearance of negative concentration, which is physically impossible. For an unstable simulation, the minimum concentration of A will grows to negative infinity, which will yields NaNs (Not a Number) when calculating concentration. Same can be said for B. In the following simulations, I will treat the occurrence of negative concentration and NaN as a sign of numerical instability.

To justify my observation, I conduct 100 simulations on different combination of Δt and h. I fix the rest of the parameters to be $k_1 = k2 = .23$, $D_A = D_B = D_{AB} = 0.0002$. Since Δt is changing, instead of duration, I fix the total time step to be $N = 2000$, which proves to be enough for us to tell if a simulation with a pair of $(\Delta t, h)$ is numerically stable or not. A and B are initialized by setting the concentration in each small square of the grid to be 5 times a random number in $[0, 1]$ (i.e. $5N$ where N obeys uniform distribution on $[0, 1]$). Figure 4 shows the result.

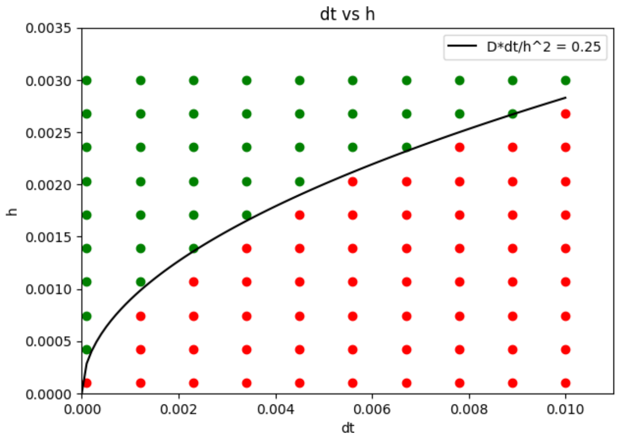

Figure 4: Plot of dt-h. Green points corresponds to simulations with low standard deviations and no overflow; Red points means the simulations encounter overflow

The green points represents the combinations of $(\Delta t, h)$ under which $[A^*]$ and $[B^*]$ of the simulations have low standard deviations and no negative value or overflow occurs during the simulation. Simulations using the combinations of $(\Delta t, h)$ at the red points encounter overflow. This result is in line with my previous observation because $(\Delta t, h)$ the area above the curve $\mu := D_X \Delta t / h^2 = 0.25$ has $\mu \leq 0.25$. Those in the area below the curve has $\mu > 0.25$

Experiment 2: Boundary of reaction An interesting phenomenon is that when the reaction is irreversible, sometimes there will be some form of blurry yet identifiable boundary between A and B during the reaction (to simulate an irreversible reaction, one can simply set $k_2 = 0$). Note that since the reaction is irreversible, A and B will eventually be exhausted. Thus, the boundary, if exists, will eventually disappear. As a result, for the following experiments, I will tune the simulation duration (by tuning the total number of time steps N [1]) so that the boundary is **visually identifiable** and

[1]In the following experiments, I will only tune the total time steps N and the initial placement of A and B. The rest of the parameters are set to be $k_1 = 0.23$, $k_2 = 0$ (so that this is an irreversible reaction), $D_A = D_B = D_{AB} = 0.0002$,

relatively stable. For convenience, instead of fixing the size of the square, I will fix the number of grids to be 200*200. The result is essentially the same up to some scaling.

Qualitatively, a boundary is relatively stable if it is visually identifiable for a period of time long enough so that we can distinguish this boundary from what may seems as a boundary but will disappear in a short amount of time (say, 2 time step). However, this period of time is always finite because A and B will eventually be exhausted.

To illustrate this point, let's look at an example. In figure 5, I place A and B in two overlapping squares. The concentration inside the square are all ones. After 1000 steps, there appears a clear boundary between A and B (marked the straight red lines in 5).

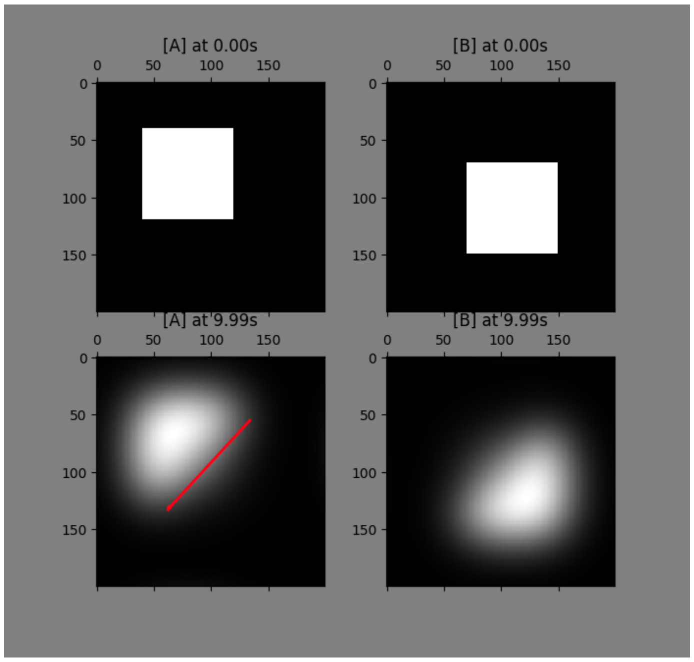

Figure 5: The distribution of $[A]$ and $[B]$ on the square. Positive concentration is represented by white color. The closer to 0, the darker the color is. A and B are initially placed in two overlapping squares. The two figures below are the results after $N = 1000$ steps.

As we increase the time steps, we can see that this boundary stays still. In figure 6, I test for $N = 2000, 5000, 10000$. We can see that although the diffusion blurs the boundary, it is still visually identifiable.

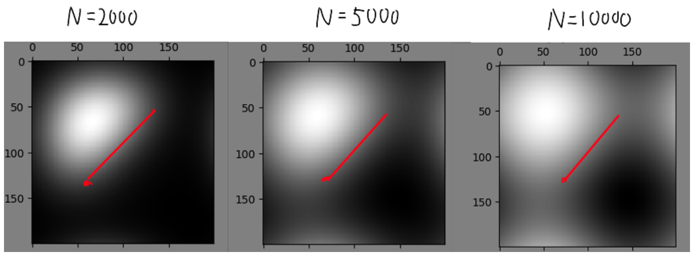

Figure 6: The distribution of $[A]$ for different values of N. Though N is increasing, the boundary marked by the red line is still visually identifiable

Figure 7 and 8 give another example. A and B are placed in two adjacent "tetris blocks". The boundary is an obtuse angle. What's different is that the boundary doesn't stay still. Instead, the degree of this angle increases as the time step increases. Eventually, it becomes a straight line across the torus.

From these simulations, we can find out some qualitative properties of the formation of boundary

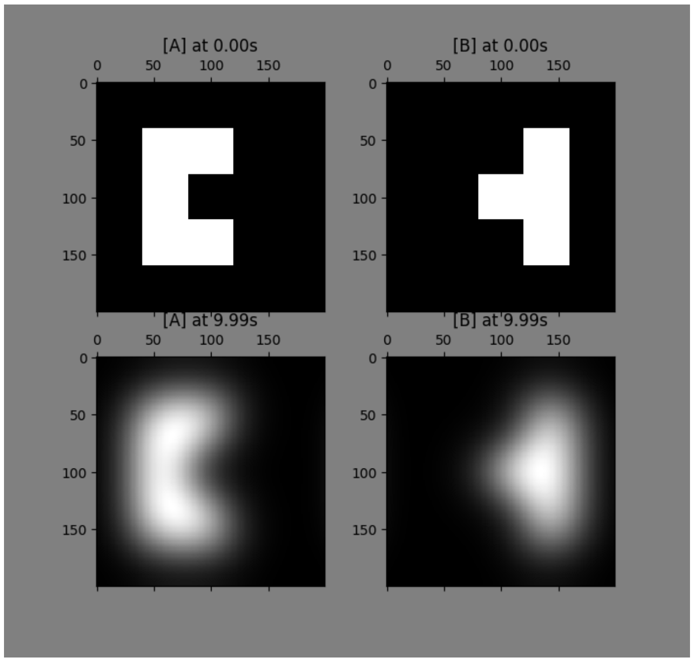

Figure 7: A and B are initially placed at two adjacent "Tetris blocks"

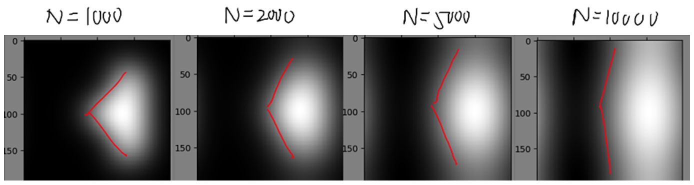

Figure 8: The distrbution of $[A]$ for different values of N. The boundary is

- To form a relatively clear boundary, A and B should be close enough (adjacent or overlapping). If A and B are far away, when they meet, their concentration will be so low due to diffusion that the boundary is not visible;
- The boundary is formed by reaction, and smoothed by diffusion.
- The boundary is harder to form if the reaction is reversible, This is because reversible reaction will further smooth the boundary

6 Summary and Conclusions

In this report, I studied a bi-molecular reaction-diffusion system on a torus. First, I defined the problem. Next, I stated and explained the system of differential equation that governs the reaction and diffusion. I proved that the quantities $[A^*]$ and $[B^*]$ are conserved in this system. Then, I discretized the system of differential equations using Euler's method. I proved a discrete approximation of Laplacian of a function with an error of $O(h^2)$. After that, I validated the correctness of my algorithm by showing that the quantities $[A^*]$ and $[B^*]$ are conserved during the simulation. I quantified this property using the standard deviation of $[A^*]$ and $[B^*]$. Finally, I conducted two experiments. In the first experiment, I examined the numerical stability of my program. Empirically, it shows that a necessary condition for numerical stability of my program is to keep the quantity $\mu = D_X \Delta t/h^2$ no larger than 0.25. In the second experiment, I explored the phenomenon of boundary between species during irreversible reaction. I tried out different initializations and found out some qualitataive properties of the boundary.

References

[Fife(1979)] Paul C. Fife. 1979. *Mathematical Aspects of Reacting and Diffusing Systems*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-93111-6

[Tornber(2011)] Anna-Karin Tornber. 2011. Lecture note from the class *Mathematical Models, Analysis and, Simulation* in Fall semester, 2011. https://www.csc.kth.se/utbildning/kth/kurser/DN2266/matmod11/PDE2_2p.pdf

A Code

This appendix presents the code for Euler method that solves the differential equation mentioned in section 2. The code is written in Python using the python packages `numpy` and `scipy` for computation, and `matplotlib` for visualization. For a full version of the code, you may refer to this github repo `https://github.com/Xinyu-Li-123/ReactionDiffusion`.

```
def initialize():
    ## Initialize
    species = np.zeros(3, dtype=float)    # [A, B, AB]
    index2name = ["A", "B", "AB"]
    # reaction coefficients
    k1 = .23
    k2 = .6
    # diffusion coefficients of A, B, AB at x and y direction, a value in [0, 1]
    D_A = 0.0002
    D_B = 0.0002
    D_AB = 0.0002

    # Define a 2d rectangle T. We will wrap it into a torus in the algorithm
    # T is a 3-dim array. Fix time t, T[0, x1, x2] is the concentration of species[0]
    #   at location (x1, x2)
    T_size = (200, 200)
    h = 0.0029        # gap of the grid, the smaller h, the more refined the grid is.
    T = np.zeros((len(species), *T_size), dtype=float)

    # Overlapping square
    T[0, 50:130, 40:120] = 1
    T[1, 70:150, 70:150] = 1

    ## Numerical solution setting
    dt = 0.01    # second
    interval = 10    # interval between frames, milisecond
    playback_speed = dt * 1000 / interval
    total_frame = 2000

    laplacian_matrix = 1 / h**2 * np.array([
        [0, 1, 0],
        [1, -4, 1],
        [0, 1, 0],
    ])

    return species, index2name, k1, k2, D_A, D_B, D_AB, T_size, h, T, save_animation, a

def simulate_without_animation(species, index2name, k1, k2, D_A, D_B, D_AB, T_size, h,
    """
    return if has overflow
    """
    # global species, index2name, k1, k2, D_A, D_B, D_AB, T_size, h, T, save_animation,

    def update_without_animation(frame_index):

        # change caused by A + B -> AB
        try:
            delta_reaction = -k1*(T[0,:,:]*T[1,:,:]) + k2*T[2,:,:]
            delta_diffusion = np.zeros((len(species), *T_size))
            for index in range(len(species)):
                delta_diffusion[index,:,:] = convolve(T[index,:,:], laplacian_matrix, m

            T[0,:,:] += ( delta_reaction + D_A  * delta_diffusion[0, :, :]) * dt
            T[1,:,:] += ( delta_reaction + D_B  * delta_diffusion[1, :, :]) * dt
            T[2,:,:] += (-delta_reaction + D_AB * delta_diffusion[2, :, :]) * dt

        except RuntimeWarning as e:
            print(e)
            return True


    print("Simulating without animation...")

    has_overflow = False

    # Codes for visualization are ignored for simplicty

    return False
```