Morphogenesys

1 Assignment Description

These assignment sheets describe the problem of morphogenesis: the process which leads to the formation patterns in living systems, like zebra stripes or our own skeleton.

The assignment consists in writing an essay that contains an answer to each question and which expand on the material included in these notes. The references are there to provide you with extra source of information and they are not exhaustive. When answering mathematical derivations, provide enough details so that we can asses that you fully understand them (They must be understandable by your peers without needing to do any derivation on the side).

The essay must be readable on its own without reference to the assignment sheets. The essay must not necessarily follow the same order as the notes and does not need to answer the questions in the same order either. As a matter of fact using a different structure and order is a bonus. Do describe the interpretation or consequences of the results that you obtain. We also expect you to focus more on some aspects of the material presented in these notes. This could be a more detailed discussion of the derivation of the equations or algorithm described in the notes or to better explain the interpretation of the results. You can also solve problems which are not set and which answer some further questions that you might have.

Some part of the assignment sheet will need to be included in your essay, but do not copy these section verbatim, use your own words. The equations do not need to be changed, but you can provide more detailed explanations or derivations when appropriate.

You have to submit 3 files: 1 pdf file for the essay and 2 python programs. We ask that you typeset your essay in LaTeX, using the provided template file and LaTeX style file.

2 Introduction

Morphogenesis [3] is the process which leads to the formation of pattern like zebra stripes, leopard's dots, crocodile skin pattern or even our skeleton. These patterns form at an early stage in the embryos and are the result of the combined diffusion of chemical substances and their mutual reactions with each other.

To create the colour patterns on the skins of animals like zebras, special cells, called melanoblasts, migrate to the surface of the embryo to become eventually pigment cells, Pigment cells generates melanin which passes into hair to give them their colour [3]. The amount of melanin that a pigment cell produces depends on the presence of a still unknown chemical which we will call X. This chemical exhibits non-uniform concentration patterns which induces the coat pattern on the animal fur. If the concentration X is large in some area, the hair will contain a large amount of pigment, while in area where it is small, the hair will contain a small amount of pigment. To generate a pattern like zebra stripes, nature has developed a system, which we call reaction-diffusion, to induce non uniform concentration of chemicals. This is what we will study in this assignment. In this assignment we analyse a simple model which exhibits most of the properties of the more realistic and more complex models.

3 Diffusion

Diffusion is the process by which small particles move randomly in their environment like, for examples, molecules in a fluid. This is described by the following equation

$$\frac{\partial u}{dt} = D\frac{\partial^2 u}{\partial x^2}, \tag{1}$$

where u is the density of the material considered and D is a parameters called the diffusion constant. The larger D is, the faster the particles diffuse. We know from experience that a drop of ink in a water glass will slowly spread itself into the water until the water gains a uniform colour. This process is very well described by the diffusion equation (1).

This means that, as a process, diffusion is precisely what we do not what for pattern formation. We must thus have something better and this come via the interaction between 2, or more, different chemicals which diffuse freely, but also interact with each other to affect their mutual concentrations. This process is called reaction diffusion. Most real life biological reaction diffusion systems are 2 dimensional (animal skins), or 3-dimensional (skeleton formation.) For simplicity we will restrict ourselves to a 1-dimensional system which, even if not describing a specific biological system, captures the essence of the process which leads to real pattern formations.

4 Reaction Diffusion Equation

In the system we are interested in, molecules diffuse but also react chemically with each other in such a way that their concentration can vary. For example, if we consider a system made out of 2 chemicals with concentration u and v and with respective diffusion constants D_u and D_v, we can have a system obeying the following general equation:

$$\begin{aligned}\frac{\partial u}{\partial t} &= f_1(u,v) + D_u \frac{\partial^2 u}{\partial x^2}\\ \frac{\partial v}{\partial t} &= f_2(u,v) + D_v \frac{\partial^2 v}{\partial x^2},\end{aligned} \tag{2}$$

where f_1 and f_2 are 2 functions which describe the interaction between the 2 chemicals. One such model is the Brusselator model for which

$$\begin{aligned}f_1(u,v) &= a - (b+1)u + u^2 v\\ f_2(u,v) &= bu - u^2 v,\end{aligned} \tag{3}$$

where $a \geq 0$, $b \geq 0$.

This system does not describe any real system as such, but it is a simple example of a model which exhibits pattern formation. Real models can be much more complex but the Brusselator captures the important feature of these models, hence our choice to study it.

The homogeneous stationary solutions of (2) are solutions which are independent of x and t and so are given by $f_1(u_0,v_0) = f_2(u_0,v_0) = 0$.

Question 1:

Homogeneous static solutions are special solutions which depend neither on t nor x. Compute these special solutions for the Brusselator, (2,3), calling them u_0 and v_0 and express them as functions of a and b. (You can ignore the trivial solution $u = 0$).

Check your solution by substituting it into the equations.

5 Stability Analysis

Having found a simple solutions, the next question is to determine if they are stable or not. The problem is the following: assume we perturb the static solution by a small amount, what does the solution do? Does it stay close to the static solution, possibly returning to the static solution after some time or does it moves away from it, never coming back. The first example is what happen to a pendulum pushed gently away from its rest position: it oscillates with decreasing amplitudes, because of friction, and ultimately returns to its static position. The second example is that of a ball pushed from the top of a hill, it accelerate and moves away from its original position never coming back.

To study the stability problem for the system (2,3) one performs what is called a linear stability analysis by assuming solutions of the form

$$\begin{aligned} u &= u_0 + \delta u(x,t) \\ v &= v_0 + \delta v(x,t). \end{aligned} \tag{4}$$

where u_0 and v_0 are the stationary solutions computed in question 1.

Question 2:

Compute the small perturbation equation for the Brusselator equations by substituting (4) into (2), with (3). Use for u_0 and v_0 the solutions obtained in question 1. You can ignore, *i.e.* disregard, any term of the type $\delta u^p(x,t)\delta v^q(x,t)$ with $p+q>1$.

Show that the resulting equation can be written in the form

$$\frac{\partial \delta \mathbf{u}}{\partial t} = \mathbf{A}\delta\mathbf{u} + \mathbf{D}\frac{\partial^2 \delta \mathbf{u}}{\partial x^2} \tag{5}$$

with

$$\delta\mathbf{u} = \begin{pmatrix} \delta u \\ \delta v \end{pmatrix}, \qquad \mathbf{A} = \begin{pmatrix} b-1 & a^2 \\ -b & -a^2 \end{pmatrix} \qquad \text{and} \qquad \mathbf{D} = \begin{pmatrix} D_u & 0 \\ 0 & D_v \end{pmatrix}. \tag{6}$$

5.1 Homogeneous Stability of Solutions

To study the stability of the Brusselator model, we must compute the solutions of equation (5,6). The simplest case consists in solutions which do not depend on x as the term proportional to $\mathbf{D}$ in equation (5) vanishes and this reduces the problem to solving a pair of ordinary differential equations

$$\frac{\partial \delta \mathbf{u}}{\partial t} = \mathbf{A}\delta\mathbf{u}. \tag{7}$$

We solve this by considering solutions of the form

$$\delta\mathbf{u} = \mathbf{w}\exp(\sigma t) \tag{8}$$

where $\mathbf{w}$ is a constant vector of arbitrary amplitude and $\sigma \in \mathbb{C}$ a constant that we must determine. As $\mathbf{w}$ defines a specific direction, its length is arbitrary, because (5) is linear, and we can impose $|\mathbf{w}| = \mathbf{1}$. Substituting (8) into (5) we obtain

$$\mathbf{A}\mathbf{w} = \sigma\mathbf{w} \tag{9}$$

showing that σ is an eigenvalue of the matrix $\mathbf{A}$. For solution to exist we must then impose the condition

$$\det(\mathbf{A} - \mathbb{1}\sigma) = 0. \tag{10}$$

Question 3:

Show that, for any matrix $\mathbf{A}$, there are 2 solutions of (10) for σ and that they are given by

$$\sigma^{\pm} = \frac{1}{2}\left(\mathrm{tr}\mathbf{A} \pm \sqrt{(\mathrm{tr}\mathbf{A})^2 - 4\det\mathbf{A}}\right). \tag{11}$$

where $\mathrm{tr}\mathbf{A}$ stands for the trace of $\mathbf{A}$ and $\det\mathbf{A}$ for its determinant. Hint : expand (10), $\det\mathbf{A}$ and $\mathrm{tr}\mathbf{A}$ in terms of $\mathbf{A}_{i,j}$ and match the different expressions.

The general linear solutions of (7) are then be given by

$$\delta\mathbf{u} = C^+\mathbf{w}^+\exp(\sigma^+ t) + C^-\mathbf{w}^-\exp(\sigma^- t) \tag{12}$$

where C^+ and C^- are complex coefficients which must be chosen so that $\delta\mathbf{u}$ is real and $\mathbf{w}^+$, $\mathbf{w}^-$ the solutions of (9) corresponding to σ^+ and σ^-.

From (12) we see that

- If σ^+ and σ^- are both real and negative, then $\delta\mathbf{u}$ decreases exponentially to 0 and the static solution u_0, v_0 is stable as the perturbed solution converges towards it.
- If either σ^+ or σ^- are positive, then $\delta\mathbf{u}$ blows up and the static solution is unstable as any small perturbation leads to a solutions that moves away from it exponentially (at least in the linear limit we have taken).
- If σ^+ and σ^- are complex, then they must be complex conjugated of each other. The imaginary part will introduce trigonometric functions which corresponds to rotations in the $u - v$ plane. The stability is again determined by the real part of $\sigma^\pm$. If it is negative, $\delta\mathbf{u}$ converges exponentially to 0 while if it is positive, it diverges.

Question 4:

1. Using (12) and (11) show that the conditions for the homogeneous static solutions to be stable are in general

$$\mathrm{tr}\mathbf{A} < 0 \tag{13}$$
$$\det\mathbf{A} > 0. \tag{14}$$

2. What do these conditions translates into for the Brusselator in terms of a and b?

One limitation of our analysis is that there is no guarantee that the ansatz (4) remains good. Indeed, when the solution is unstable, δu or δv can become very large and break down our initial assumption. To find out what the real solution does we can write a computer program that solves the system of equation (2) with (3) and take $D_u = D_v = 0$ which is equivalent to assuming that u and v are independent of x. This leads to the following pair of ordinary differential equations

$$\begin{aligned} \frac{\partial u}{\partial t} &= a - (b+1)u + u^2 v \\ \frac{\partial v}{\partial t} &= bu - u^2 v \end{aligned} \tag{15}$$

which can be easily solved numerically.

Coding task 1:

Write a computer program called `bxl_ode.py` to solve equation (15) using the class `ODE_RK4` provided in the module `ode_rk4.py` (this is the same module as the one used during the practicals). The program `lotka_volterra_rk4.py`, which is a cleaned up version of a program which you wrote during the practicals, can be used as a template (don't forget to update the comments).

- The program must define a class called `Bxl_ODE` as a subclass of `ODE_Rk4`
- The class must contain an initialiser `__init__(self,V0=[0], dt=0.1, t0=0,a=1,b=1)`, which must set the class variables `a` and `b`. (This is very similar to `lotka_volterra_rk4.py`).
- The class must contain the class function `F(self,t,V)` which returns the right hand side of equation (15) as a 2 element `numpy` array. It must use the class variable `a` and `b`. (This is very similar to `lotka_volterra_rk4.py`).
- The class must include a function called `static_solution(self)` which takes no argument and computes the homogeneous static solution of the Brusselator, as computed in Question 1, and return `[u0,v0]` as a `numpy` array.

- The class must include a function called `sigma(self)` which computes the 2 stability eigenvalues σ^+ and σ^- given by equation (11). It must also compute the value $\mathrm{tr}\mathbf{A}$ and the discriminant $\mathrm{disc} = \mathrm{tr}\mathbf{A}^2 - 4\det\mathbf{A}$ where $\mathbf{A}$ is given by (6). The function must return σ^+ and σ^-, $\mathrm{tr}\mathbf{A}$ and disc as a tuple in that order.
- The following code defines the function `plot_u()` which generates a figure of $u(t)$ using the function `plot` of the class `ODE_Rk4`. `title` is the title for the figure and `fname` a filename used to save the figure in a file (if `fname` is `""` the figure is not saved.)

```python
def plot_u(self,title="",fname=""):
    """ display  u(t)
    :param title: the title for the figure
    :param fname: the ouput filename, including the file
        extension.
                  The figure is not saved if fname is ""
    """
    self.plot(1,0,style="k-")
    plt.title(title,fontsize=16)
    plt.xlabel("t",fontsize=22) # Set horizontal (x) figure label
        to "t"
    plt.ylabel("u",fontsize=22) # Set vertical (y) figure label
        to "u"
    if fname!="" : plt.savefig(fname)
    plt.show()
```

  Add the definition of `plot_u()` into the `Bxl_ODE` class definition. Add another 2 similar class functions called `plot_v` and `plot_v_u`, both taking the same argument as `plot_u` and which plot respectively $v(t)$ and $u(v)$. The figures $u(t)$ appears as a black curve. We ask that $v(t)$ is a blue curve and $u(v)$ a series of red stars (use the `style` argument of the `ODE_Rk4` `plot` function to set these curve parameters). The `plot` function is defined in the module `ode_rk4.py` and was described on page 7 of the *Continuous time models* problem sheet. Its use is also illustrated in the file `lotka_volterra_rk4.py`.
- Your code will be tested using the supplied program `run_bxl_ode.py` which illustrate how to set the different parameters.

Question 5:

1. Taking $a = 2$, find the following:
   - 1) A value of b for which $\sigma^+ \neq \sigma^-$ are both real and negative.
   - 2) The smallest value of b for which $\sigma^+ = \sigma^- < 0$.
   - 3) A value of b for which $\sigma^+ \neq \sigma^-$ are complex with a negative real value.
   - 4) The value of b for which $\sigma^+ = -\sigma^-$ are both complex.
   - 5) A value of b for which $\sigma^+ \neq \sigma^-$ are complex with a positive real value.

   Start by computing algebraically the values of b for bullet point 2 and 4 and use these 2 results to find the others values.
2. Use these values to fill in the following table

a	b	u_0	v_0	$\mathrm{tr}\mathbf{A}$	$\sqrt{\mathrm{tr}\mathbf{A}^2 - 4\det\mathbf{A}}$	σ^+	σ^-
2	(b.1)						
2	(b.2)						
2	(b.3)						
2	(b.4)						
2	(b.5)						

(The values in the table should be evaluated numerically, using your program for example.)

3. Generate the figures $u(t)$ (black curve), $v(t)$ (blue curve) and $u(v)$ (red stars) for the parameter values $a = 2$ and the last 3 values of b (b.3, b.4 and b.5) selected from the table above. As an initial condition use $u(0) = u_0 + \epsilon$ and $v(0) = v_0 + \epsilon$ with $\epsilon = 0.1$ (or a smaller value if it leads to better figures. ϵ does not need to be the same for all figures). The integration time `tmax` will need to be adjusted. Start with `tmax=100` and then adjust it to best illustrate what the solution does. Use $dt = 10^{-4}$.
4. Comment each of the solutions obtained for the values of b that you have selected for the table above (b.1 to b.5). Which one are stable and which one are not? Are unstable solutions unstable even for very small ϵ? Are the prediction of the stability analysis confirmed by the numerical solutions?

So far we have only considered perturbations which are independent of x, so we will now consider the more general case of non homogeneous perturbations.

5.2 General Stability Analysis

To perform a more general stability analysis of solutions of (2, 3) one considers perturbations $\delta u(x,t)$ and $\delta v(x,t)$ which depend both on t and x. This is described in detail in the Appendix where it is shown that for an homogeneous static solution to be stable under general perturbation the following condition must be satisfied:

$$\begin{aligned} \mathrm{Tr}\mathbf{A} &< 0 \\ D_v\mathbf{A}_{11} + D_u\mathbf{A}_{22} &< 2\sqrt{D_u D_v(\mathbf{A}_{11}\mathbf{A}_{22} - \mathbf{A}_{12}\mathbf{A}_{21})}. \end{aligned} \tag{16}$$

If $\det\mathbf{A} < 0$ then condition (33) can't be evaluated because the right hand side is purely imaginary, but in that case, we know, from the first part of our study, that the solution is unstable under homogeneous perturbations. As shown for homogeneous perturbation, if $\mathrm{Tr}\mathbf{A} > 0$ the solutions is always unstable.

Note: For your essay you can simply state *it can be shown ...* quoting the assignment sheet as a reference. You can also include the content of the appendix and details some of its derivations, to score points on the essay depth.

6 Numerical Integration of the Brusselator

To better understand the properties of the Brusselator we can write a computer program to solve equations (2) with (3). To do this we must transform equation (2) into a discrete system of equations. Using the fact that for any function $u(x)$,

$$\frac{d^2u}{dx^2} \approx \frac{u(x+\Delta x) + u(x-\Delta x) - 2u(x)}{\Delta x^2} \tag{17}$$

we can approximate (2) by

$$\begin{aligned} \frac{du_i}{dt} &= D_u\frac{u_{i+1} + u_{i-1} - 2u_i}{\Delta x^2} + f_1(u_i, v_i) \\ \frac{dv_i}{dt} &= D_v\frac{v_{i+1} + v_{i-1} - 2v_i}{\Delta x^2} + f_2(u_i, v_i). \end{aligned} \tag{18}$$

The program `brusselator_pde.py` solves (2) with (3) numerically by solving (18) using the 4th order Runge-Kutta method to perform the time integration. The boundary conditions that we use are that $\partial u/\partial x$ and $\partial v/\partial x$ vanish on the edges of the domain.

Look at the file `run_brusselator_pde.py` while reading the description of `brusselator_pde.py`

`brusselator_pde.py` is a complete program in which you will only have to complete 2 lines (see below), [because we didn't want to give you the solution of question 1]. The program `run_brusselator_pde.py` shows you how to use `brusselator_pde.py`. It defines a class called `BrusselatorPDE` and its initialiser takes 7 arguments: `L` the length of the domain which we will set to `50`, `N` the number of grid points used to solve the discretised equation, taking `100` for our investigations, `t0` the integration initial time which will set to `0` and finally `Du`, `Dv`, `a` and `b` which are the model parameters.

To solve the equation, one must first call the class function `initial_condition()` which computes $u(x,0) = u_0 + \epsilon \exp(-(x-0.5\,L)^2$, $v(x,0) = v_0 + \epsilon \exp(-(x-0.5\,L)^2$. u_0 and v_0 are the homogeneous static solution which you computed in question 1 and the exponential term is a small perturbation used to check the stability of the homogeneous static solutions. The parameter ϵ controls the size of the perturbation and we will mostly use $\epsilon = 0.01$.

One then calls the class function `iterate(tmax,fig_dt)` to integrate the equation up to `tmax`. `fig_dt` controls how often the intermediate values of u and v are saved to later plot function profiles (we will use `tmax/100`).

If `br` is the variable containing the `BrusselatorPDE`, after calling the function `br.iterate()` the values of $u(x)$ and $v(x)$ are available in the class variable `br.v` as a numpy array such that $u_i = u(x_i)$ =`br.v[2*i]` and $v_i = v(x_i)$ =`br.v[2*i+1]` where $x_i = i\,L/(N-1)$.

Before solving the next coding task, you must do the following:

- Complete line 26 and 27 of `brusselator_pde.py` setting the class variable `u0` and `v0` to correspond to the homogeneous static solutions computed in question 1. (The code does not need to be returned).

- Use the program `run_brusselator_pde.py` using `espilon=0` to check that your homogeneous constant solution is correct.

- Run `run_brusselator_pde.py` with `espilon=0.01` for different values of `b` in the range [1,6], keeping the other parameters unchanged. Notice the differences between stable and unstable solutions.

- Increase `tmax` to convince yourself that the unstable solutions ultimately settle to other static but non homogeneous solutions $u(x,t)$ and $v(x,t)$ which depends on x but not on t.

- Do you understand why we are interested in unstable solutions for pattern formation? Can you see the link between the solution $u(x)$ (or $v(x)$) and zebra stripes?

We would now like to study the solutions more systematically and for this we need to determine when the solutions are unstable. From the solution we have just computed, the difference is obvious. Stable homogeneous static solutions are, as the name implies, constant, while unstable solutions evolve to other solutions which exhibits some local maxima and minima. This is easy to detect, as all we have to do it to compute the maximum and minimum values of u. The difference between the two correspond to the amplitude of spacial oscillation of the solution:

$$A_u = \max(u(x)) - \min(u(x)), \qquad A_v = \max(v(x)) - \min(v(x)). \tag{19}$$

We will now write a program that will integrate the equation for a range of values for b and generate figure of A_u and A_v as function of b.

Coding task 2:

The first part of the coding task will consists in writing a program called `bxl_analysis.py`.

- `bxl_analysis.py` must first import `BrusselatorPDE` from `brusselator_pde.py` as well as `numpy` and `matplotlib.pyplot`.
- As in `run_brusselator_pde.py`, set the following parameter values: `a = 1.5`, `b = 3`, `Du = 2.8`, `Dv = 22.4`, `L = 50`, `N = 100`, `tmax = 100`, `fig_dt=tmax/100`, `epsilon = 0.01`,
- Write a function `amp(b,Du)` which compute the amplitudes A_u and A_v.
  - It must first create a `BrusselatorPDE` object, and call its class function `initial_condition(epsilon)` and `iterate(tmax, fig_dt)`.
  - It must then compute the maximum and minimum value of $u(x)$ to compute A_u and similarly for A_v for $v(x)$, (19) (see the above description of how u and v are stored as a numpy array in the class variable `v`). Then use the numpy function `amax` and `amin` to compute the extrema and compute the amplitudes from these numbers.

  - The function must then return A_u and A_v as a tuple.
- Write a loop which scans value of b between 0.5 and 5, for 50 different values (but set this to 5 when you test your code). The loop must simply call the function `amp(b)` for each value of b and generate a single figure showing the values of A_u and A_v as functions of b. The values ofA_u must be plotted as black `*` and the values of A_v as blue `+`.
- When the solutions are stable, the `*` and the `+` will overlap making one of then invisible. To overcome this problem, subtract `0.05` from the values of `b` used to plot A_v. This will shift the `+` slightly to the left and make them visible.
- Add a label to your figure: the horizontal label must be b and the vertical one A_u, A_v. (For best looking results you can use the latex python features simply using something like `s = r'$x_i^n$'` which produces x_i^n when using `s`.)
- Add a title to you figure. It must contain the values of the parameters a, D_u and D_v.
- Use the pyplot `savefig` function to save your figure into a file. The filename must contain a sensible name including the values of the parameters a, D_u and D_v (so that you know what each file is).
- So far the program generates 1 figure showing A_u and A_v as function of b. We now want to generate separate figures for different values of D_u. To do this add a loop which scans 7 values of D_u for values in the range 1 to 7 so that it generates 1 figure for each of these values.
- The full program will take a little while to run. To avoid having to close the figures when they are generated, you can replace the line `plt.show()` by `plt.close()`. Your program will then generate the figures as files in the background allowing you to do something else during that time.

Run your program and check from which value of b the homogeneous static solutions become unstable. How does this compare with our prediction?

In the second part of the coding task, we will improve our code so that it displays A_u and A_v using different colours based on our theoretical prediction that the solution is stable or not.

- As we will plot the `*` and `+` in different colours based on the stability condition, the first step is to modify your program `bxl_analysis.py` so that it plots each of these symbols, one at a time, as soon as the values for A_u and A_v have been computed.
- Inside the loop scanning values of `b`, set a boolean variable `stable` to `True` if the stability condition in section 5.2 is satisfied and `False` otherwise.
- Modify your program so that it plots A_u as black `*` when `stable` is true and as green `*` otherwise. A_v must be plotted as blue `+` when `stable` is true and as red `+` otherwise.

Question 6:

1. Provide at least 2 figures generated by `bxl_analysis.py`. Looking at these figures, describe how good the theoretical prediction is.
2. How can we interpret the profile of $u(x)$ and $v(x)$ in the context of morphogenesis?. Explain why we are seeking systems for which the homogeneous static solutions are unstable.
3. What is the expected interpretation of the parameter q_m and its wave number $n_m = Lq_m/\pi$ given by eq (27)? Hint: look at the profiles of u and v for some parameter values and compute the corresponding values of n_m. (To answer this question you must read the appendix.)
4. What difference does diffusion make to the system (comparing the 2 sections of the project)?

6 Appendix: General Stability Analysis

To perform a more general stability analysis of solutions of (2) with (3) we consider more general, space dependant, perturbations of the form

$$\delta\mathbf{u} = \mathbf{w}_q \exp(\sigma_q t)\exp(iqx), \tag{20}$$

where σ_q and q are constants which must be determined and $\mathbf{w}_q$ a constant vector. As the stability equation (5) is linear $\mathbf{w}_q$ is defined only up to an overall constant. The only difference with what we did in the previous section is that the perturbation (20) depends on x. Inserting (20) into (5) we obtain the condition

$$\left(\mathbf{A} - \mathbf{D}q^2\right)\mathbf{w}_q \quad = \quad \sigma_q \mathbf{w}_q. \tag{21}$$

For solution to exist we must impose

$$\det\left(\mathbf{A} - \mathbf{D}q^2 - \mathbb{1}\sigma_q\right) = 0. \tag{22}$$

The solutions for σ_q are similar to the one we have obtained before except that we must replace $\mathbf{A}$ by

$$\mathbf{B} = \mathbf{A} - q^2\begin{pmatrix} D_u & 0 \\ 0 & D_v \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{11} - D_u q^2 & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} - D_v q^2 \end{pmatrix}. \tag{23}$$

We then have

$$\sigma_q^{\pm} = \frac{1}{2}\left(\mathrm{tr}\mathbf{B} \pm \sqrt{(\mathrm{tr}\mathbf{B})^2 - 4\det\mathbf{B}}\right). \tag{24}$$

and it can be shown that general linear solutions will then be given by a linear superposition of all solutions of the type (20)

$$\begin{aligned}\delta\mathbf{u} \quad = \quad & \sum_q \left[\mathbf{w}_q^+ \exp(\sigma_q^+ t)(C_{+q}^+ \exp(iqx) + C_{-q}^+ \exp(-iqx))\right. \\ & \left. + \mathbf{w}_q^- \exp(\sigma_q^- t)(C_{+q}^- \exp(iqx) + C_{-q}^- \exp(-iqx))\right]. \end{aligned} \tag{25}$$

Notice that the ($C_{\pm q}^{\pm}$ are nothing but the Fourier coefficients of the of the perturbation $\delta\mathbf{u}$. To specify the solution we must also impose some boundary conditions. We will assume that we have a finite domain ranging from 0 to L and we will impose that the chemicals bounce on the edges of the domain when they reach them. This is mathematically expressed by imposing the condition

$$\left.\frac{\partial\delta\mathbf{u}}{\partial x}\right|_{x=0} = \left.\frac{\partial\delta\mathbf{u}}{\partial x}\right|_{x=L} = 0 \tag{26}$$

at all times t. As (26) must be true for all time, we can consider $\mathbf{w}_q^+ = 0$ or $\mathbf{w}_q^- = 0$ independently and as a result imposing (26) on (25) implies that $C_{+q}^+ iq\exp(i\,q\,0) - C_{-q}^+ iq\exp(-i\,q\,0) = 0$ and $C_{+q}^+ iq\exp(i\,q\,L) - C_{-q}^+ iq\exp(-i\,q\,L) = 0$ with similar conditions for the $C_{\pm q}^-$. These conditions imposes some restrictions on the values that q can have. The first condition implies $C_{+q}^+ = C_{-q}^+$ and the second condition implies $iqL = in\pi$ and so

$$q_n = \frac{n\pi}{L}. \tag{27}$$

where n is an arbitrary positive integer. From now on we will label each value of q with the index n as in (27). Similarly we also have $C_{+q}^- = C_{-q}^-$. We can thus rewrite (25) as

$$\delta\mathbf{u} \quad = \quad \sum_{n=0}^{\infty}\left[C_{q_n}^+ \mathbf{w}_{q_n}^+ \exp(\sigma_{q_n}^+ t) + C_{q_n}^- \mathbf{w}_{q_n}^- \exp(\sigma_{q_n}^- t)\right]\cos(\frac{n\pi x}{L}). \tag{28}$$

In the previous section, we assumed that the solutions did not depend on x, which is equivalent to taking $\mathbf{w}_{q_n} = 0$, $\forall n > 0$, and we obtained some criteria for the stability of the solution. All the solutions which were unstable under the homogeneous perturbation are obviously unstable but they might have more instability modes than the homogeneous one. The question we really want to answer is what happen to the solutions which are stable under the homogeneous perturbation? Are they all still stable or do some of them become unstable when we consider x dependant perturbations?

The conditions for stability now becomes

$$\begin{aligned} \mathrm{tr}\mathbf{B} &= \mathbf{A}_{11} + \mathbf{A}_{22} - (D_u + D_v)q^2 < 0 \\ \det \mathbf{B} &= (\mathbf{A}_{11} - D_u q^2)(\mathbf{A}_{22} - D_v q^2) - \mathbf{A}_{12}\mathbf{A}_{21} > 0. \end{aligned} \tag{29}$$

Given that q^2, D_u and D_v are all positive, the first condition implies that if $\mathbf{A}_{11} + \mathbf{A}_{22} < 0$, then $\mathrm{tr}\mathbf{B} < 0$ for all values of q.

The question we should answer is what are the values of the model parameters, for which $\det \mathbf{B} > 0$ for all the q_n given by (27). This is a difficult problem to solve in general, so we can simplify it by noticing that the determinant of $\mathbf{B}$ is a parabola in q^2 which is positive for large q. The minimum of the parabola is obtained when $d \det(\mathbf{B})/d(q^2) = 0$.

Using equation (29) we define q_m as the value of q for which

$$\left.\frac{d \det(\mathbf{B})}{d(q^2)}\right|_{q=q_m} = -(D_v \mathbf{A}_{11} + D_u \mathbf{A}_{22}) + 2q_m^2 D_u D_v = 0. \tag{30}$$

and so

$$q_m^2 = \frac{D_v \mathbf{A}_{11} + D_u \mathbf{A}_{22}}{2 D_u D_v}. \tag{31}$$

Then

$$\det(\mathbf{B}(q_m)) = -\frac{1}{4 D_u D_v} (D_v \mathbf{A}_{11} + D_u \mathbf{A}_{22})^2 + \mathbf{A}_{11}\mathbf{A}_{22} - \mathbf{A}_{12}\mathbf{A}_{21} \tag{32}$$

and the condition $\det(\mathbf{B}(q_m)) < 0$ implies

$$D_v \mathbf{A}_{11} + D_u \mathbf{A}_{22} > 2\sqrt{D_u D_v (\mathbf{A}_{11}\mathbf{A}_{22} - \mathbf{A}_{12}\mathbf{A}_{21})}. \tag{33}$$

For the Brusselator model, we have $\mathbf{A}_{11} > 0$ and $\mathbf{A}_{22} < 0$ and for convenience, we define the diffusion length

$$l_1 = \sqrt{\frac{D_u}{\mathbf{A}_{11}}} \qquad l_2 = \sqrt{\frac{D_v}{-\mathbf{A}_{22}}}. \tag{34}$$

We can then divide (33) by $2 D_u D_v$ and rewrite it as

$$q_m^2 = \frac{1}{2}\left(\frac{1}{l_1^2} - \frac{1}{l_2^2}\right) > \sqrt{\frac{\mathbf{A}_{11}\mathbf{A}_{22} - \mathbf{A}_{12}\mathbf{A}_{21}}{D_u D_v}}. \tag{35}$$

This means that the length l_2 must be sufficiently larger than l_1 to have instability.

The condition that $\mathbf{A}_{11} > 0$ means that the first chemical enhances its own instability while $\mathbf{A}_{22} < 0$ means that the second chemical inhibits its own growth. The first chemical is called an activator and the second an inhibition. The condition $l_2 > l_1$, called Turing condition, is interpreted as saying that the range of inhibition must be larger than the range of activation.

6 References

[1] Michael Cross. Notes on the Turing Instability and Chemical Instabilities. http://www.cmp.caltech.edu/~mcc/BNU/Notes7_2.pdf

[2] A.M. Turing. The Chemical Basis of Morphogenesis. *Philosophical Transactions of the Royal Society of. London. Series B, Biological Sciences*, Vol. 237, No.641, (1952), p 37-72.

[3] J.D. Murray. *Mathematical Biology II: Spatial Models and Biomedical Applications.* Springer; 3rd ed. 2003. (Available as an ebook from the University library)

7 To submit:

- One pdf file for the essay, including the figures for questions Ensure all your figures have axis labels which are not tiny. Give all your figures a caption describing their content and refer to them in the text by number.
- Python code 1: `bxl_ode.py`
- Python code 2: `bxl_analysis.py`