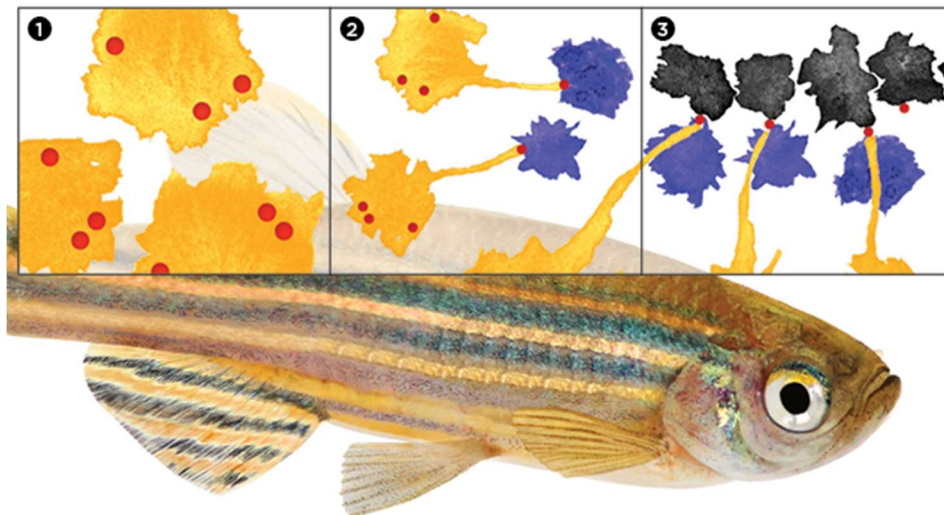


Rapport UE BS1

Pigment cell movement is not required for generation of Turing patterns in zebrafish skin

D. Bullara & Y. De Decker - 2015



Introduction

Le zebrafish comme organisme modèle

Le poisson-zèbre ou zebrafish (*Danio rerio*), est un organisme modèle très utilisé en recherche depuis les années 60, car il est facilement élevable en laboratoire et se reproduit rapidement. Il possède des caractéristiques facilitant l'étude de divers phénomènes, notamment du développement, puisque certaines souches sont translucides, ce qui permet une visualisation en direct de mécanismes morphogénétiques (par exemple le développement des organes et du système sanguin).



Figure 1. Un poisson-zèbre présentant des rayures horizontales, dont la morphogenèse est encore inconnue.

Il a ainsi un rôle dans la recherche médicale, puisque nous pouvons grâce à lui étudier les tumeurs, les maladies cardio-vasculaires et autres anomalies du développement. En recherche fondamentale, le zebrafish est utilisé pour comprendre les mécanismes sous-jacents au développement morphologique. En effet, celui-ci possède des rayures sur le corps dont le mécanisme de formation n'est pas bien défini (cf figure 1). Le mode d'apparition de ces rayures est inconnu, sont-elles dues à des cellules qui s'auto-organisent et se déplacent pour produire des motifs spécifiques, ou bien un phénomène physico-chimique induirait-il leur apparition ?

Un article publié en 2015 et rédigé par Bullara et De Decker semble répondre à cette question, en modélisant la réaction qui permet l'apparition spontanée de motifs sans intervention de mouvement cellulaire. Cette modélisation est basée sur l'hypothèse de Alan Turing selon laquelle une réaction chimique peut induire l'apparition de phénomènes spontanés. Ce modèle est repris et amélioré par Bullara et De Decker pour montrer, avec l'appui d'autres recherches, que les cellules à l'origine des rayures sur le zebrafish ne sont pas en mouvement lors de l'établissement du motif, mais que celui-ci est plutôt dû aux interactions entre cellules.

Le modèle minimal proposé par Bullara et De Decker permet de mimer des motifs observés expérimentalement, et validerait donc l'hypothèse de réaction-diffusion initialement proposée par Alan Turing.

Nous avons reproduit leurs résultats en programmant à notre tour les différentes étapes décrites dans l'article. Pour cela, nous avons choisi d'utiliser le langage Python, afin de produire un script (cf Annexe page 7) qui permet la modélisation de rayures dans une matrice suivant des motifs semblables à ceux obtenus par les auteurs.

Le modèle de Turing

Le modèle de Turing est un modèle mathématique proposé par Alan Turing en 1952 et qui explique la formation de certains motifs lors de la morphogenèse.

Selon ce modèle, la conjonction de certaines réactions chimiques et de la diffusion moléculaire des réactifs entraîne spontanément une variation spatiale des concentrations d'espèces chimiques, produisant des bandes ou des taches régulièrement espacées. Ces structures sont maintenant appelées "structures de réaction-diffusion" ou "structures de Turing". Alan Turing a suggéré que ces processus purement physico-chimiques pouvaient être à la base de la morphogenèse animale et végétale en induisant le développement de structures répétées.



Figure 2. Alan Turing
(1912-1954)

Le principe de Turing peut être décrit brièvement comme suit : une espèce chimique X va favoriser localement sa propre production, car faisant partie d'une réaction autocatalytique. Cette rétroaction chimique entraîne une activation de X à courte distance. Cette réaction provoque également l'apparition d'une seconde espèce Y qui favorise la consommation de X. Si Y diffuse plus rapidement que X, cette deuxième rétroaction chimique entraîne une inhibition à longue distance de X. La combinaison de ces deux effets induit une distribution spatiale non triviale des concentrations des différentes espèces chimiques, ce qui permet ainsi l'apparition de motifs spécifiques.

Dans le modèle de Turing se trouvent deux éléments, un activateur et un non activateur, dont l'interaction peut se traduire mathématiquement par les équations suivantes :

$$d_A/dt = \Delta_A + f(A+I)$$

$$d_I/dt = \Delta_I + f(A+I)$$

Δ représente la diffusion, f représente la loi de diffusion locale, et les espèces A et I s'influencent positivement ou négativement. Ce modèle permet donc de ramener un processus biologique (la morphogenèse) à une équation mathématique traduisant une réaction chimique qui peut se comprendre facilement. C'est cette simplification qui a fait la renommée du modèle de Turing, et motivé son utilisation dans de nombreux articles.

L'organisme modèle *Danio rerio* présente des motifs rayés qui ressemblent à ceux générés par le phénomène de réaction-diffusion du modèle de Turing. L'article de Bullara et De Decker cherche à démontrer que l'apparition des motifs des zebrafish correspond en effet à un mécanisme de Turing. Pour cela, ils ont réalisé un programme simulant la morphogenèse de motifs en fonction de différents paramètres, que nous avons testé dans

notre propre algorithme. Les motifs sont définis selon l'agencement des cellules de chromatophores, qui peuvent être de deux types : xanthophores ou mélanophores.

Modélisation

Principe du code utilisé

Le script que nous avons programmé suit plusieurs étapes décrites dans l'article. Il permet de coder sept événements biologiques ayant chacun une probabilité d'apparition spécifique :

- Naissance des cellules de xanthophores et de mélanophores (taux bx et bm) ;
- Mort des cellules de xanthophores et de mélanophores (taux dx et dm) ;
- Interaction entre les cellules à courte distance, qui augmente la probabilité de mort cellulaire, soit par action des mélanophores sur les xanthophores (taux sm), soit inversement des xanthophores sur les mélanophores (taux sx) ;
- Enfin, une interaction à longue distance qui favorise la naissance de mélanophores lorsque des xanthophores se trouvent à grande distance d'eux (taux lx, distance h).

La probabilité d'apparition d'un de ces événements est définie par les paramètres qui leur sont associés entre parenthèses. La modification de la valeur de ces paramètres permet l'obtention de différents motifs sur une matrice de taille 100*100.

L'algorithme fonctionne grâce à une boucle d'un milliard d'itérations qui permet d'essayer de produire un des sept événements sur une case de la matrice choisie aléatoirement. Le choix d'un des événements se fait également selon un calcul de probabilité.

La succession des étapes de l'algorithme est représentée en figure 3.

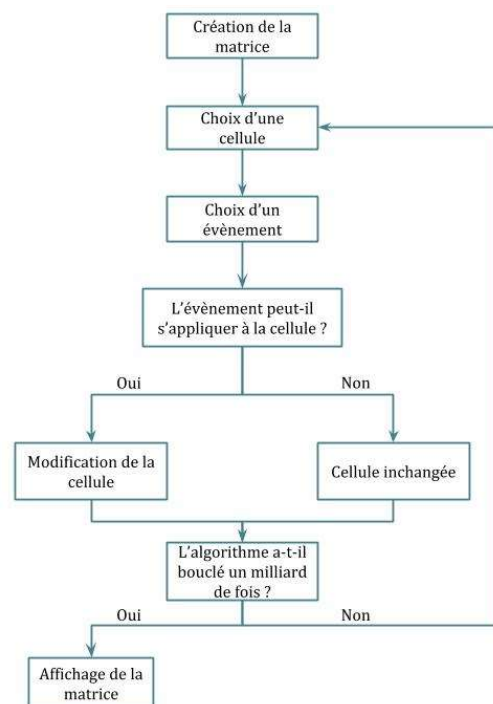


Figure 3. Représentation schématique des différentes étapes de l'algorithme.

Exemple de résultats obtenus

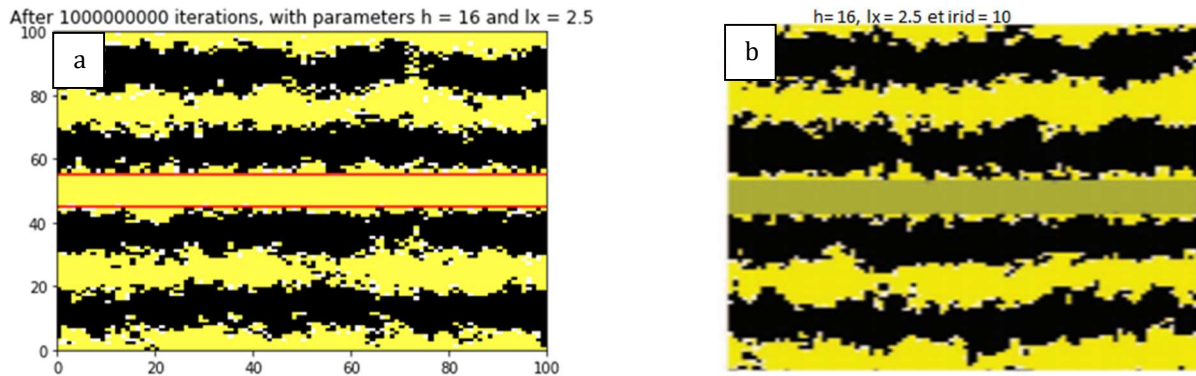


Figure 4. Comparaison des motifs obtenus avec notre algorithme (a) et celui des auteurs (b). En (a), les lignes rouges délimitent une couche d'iridophores, représentée en gris sur la figure (b).

Nous pouvons voir en figure 4 un exemple de résultat produit par notre algorithme, avec les mêmes paramètres que ceux décrits dans l'article. On remarque que nous avons pu obtenir des régions distinctes en fonction de la couleur, qui représente les mélanophores (noir) et les xanthophores (jaune). La valeur du paramètre lx nous a permis d'avoir un motif homogène, une valeur inférieure nous aurait donné un motif totalement jaune (composé uniquement de xanthophores) tandis qu'une valeur plus élevée nous aurait donné un fond noir avec des points jaunes, comme démontré par les auteurs. La valeur du paramètre h nous a permis d'avoir des mélanophores situés à une distance h des xanthophores, plus cette valeur diminue et plus la largeur des rayures est petite.

Modification des paramètres et conséquences sur le motif obtenu

Taux d'interaction à longue distance lx

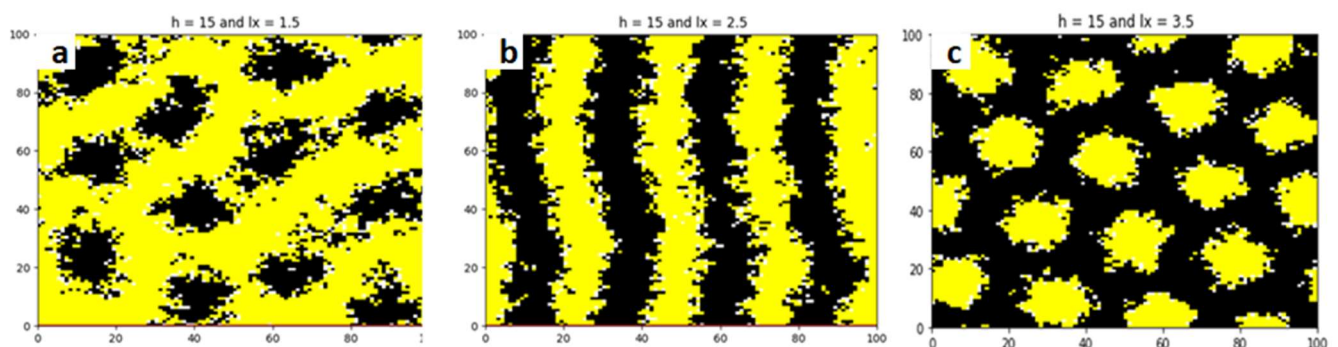


Figure 5. Motifs obtenus en utilisant différentes valeurs de lx à partir de notre algorithme.

Comme nous avons pu le voir à travers la figure 5, le paramètre lx permet de contrôler la géométrie du motif. Lorsque l'interaction à longue distance possède un taux lx élevé, l'apparition de mélanophores est amplifiée lorsqu'un xanthophore est éloigné d'une distance h . Ceci produit un motif dominé par des mélanophores (figure 5.c). Au contraire, si le taux lx diminue, le motif produit sera dominé par des xanthophores (figure 5.a). Ainsi, des valeurs extrêmes de lx permettent la prolifération d'un type particulier de chromatophores, pour un taux faible, les xanthophores représentent la majorité des cellules formant le motif, alors qu'avec un taux élevé, ce sont les mélanophores qui sont majoritaires. Une valeur intermédiaire de lx permet d'obtenir un motif homogène, avec une répartition égale en xanthophores et mélanophores. Cet équilibre est obtenu avec une valeur $lx = 2,5$.

Effet de la variation de la distance h (interaction à longue distance)

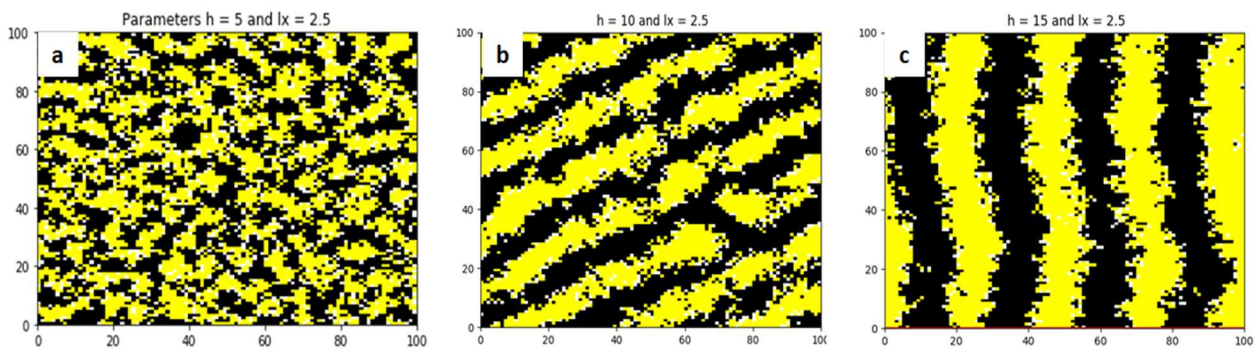


Figure 6. Motifs obtenus avec différentes valeurs de h , pour le même taux lx .

D'après la figure 6, on remarque que la largeur des rayures est affectée directement par le paramètre h (rayon d'action de l'interaction à longue distance), puisque la valeur de lx est constante. La distance h permet d'avoir une surface entourant les xanthophores où les mélanophores ne peuvent pas naître. Ceci est bien visible avec une grande valeur de h (b, c), qui induit l'apparition des rayures distinctes, contrairement à la figure 6.a où on retrouve un mélange de cellules non organisées.

Influence des iridophores

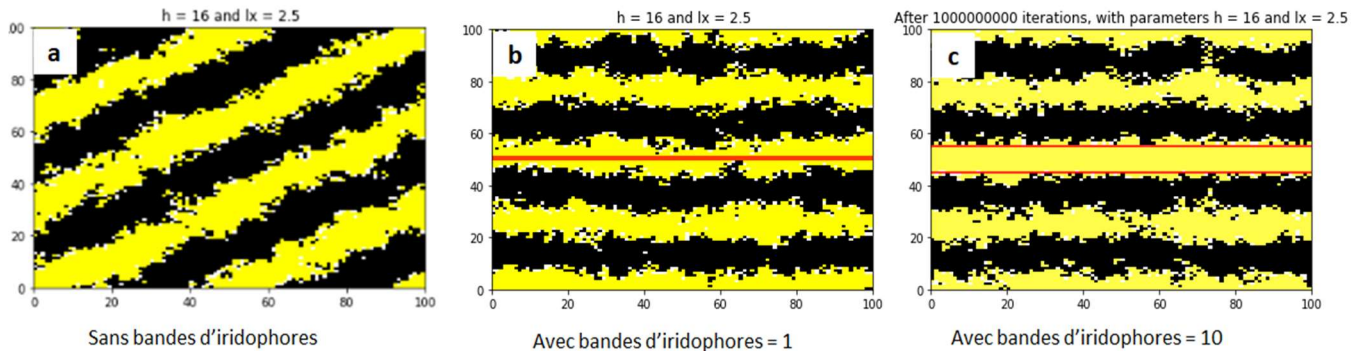


Figure 7. Motifs obtenus avec (b, c) et sans bandes d'iridophores (a).

L'ajout d'une ligne d'iridophores nous a permis d'obtenir un motif présentant des rayures horizontales. Comme on peut le voir sur la figure 7, l'initialisation de notre matrice avec une ligne d'iridophores, quel que soit sa largeur, empêche l'apparition de mélanophores sur les cellules jouxtant les iridophores. Cela force donc l'apparition d'une bande de xanthophores parallèlement à la bande d'iridophores. Le motif final sera donc rayé horizontalement.

Conclusions

Durant ce projet, nous avons pu reproduire le modèle minimal proposé par D. Bullara et Y. De Decker en 2015, et qui donne une explication au mécanisme de morphogenèse des motifs portés par les poissons-zèbres. Nous avons modélisé les diverses interactions existant entre les deux types de cellules de chromatophores (xanthophores et mélanophores) et qui permettent de produire différents motifs selon les paramètres utilisés.

Les résultats obtenus avec notre algorithme sont semblables à ceux produits par les auteurs et publiés dans leur article. Ceux-ci correspondent de plus aux observations expérimentales faites par les auteurs. Le phénomène de réaction-diffusion décrit par le modèle de Turing semble donc une bonne hypothèse concernant la formation des motifs du zebrafish.

Les cellules de chromatophores resteraient donc immobiles pendant la morphogenèse. C'est l'interaction entre ces cellules qui induirait leur apparition ou leur disparition, ce qui permettrait de modéliser la forme du motif. Ce motif peut être influencé par la présence d'iridophores inhibant les mélanophores.

L'algorithme utilisé est disponible en Annexe 1 page 7, des exemples de motifs obtenus sont présentés en Annexe 2 page 15.

Annexe 1 - Algorithme

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import numpy as np
import random
import math
import matplotlib.pyplot as plt
import matplotlib.colors as colors

##### Base functions

def initialization_with_iridophores(length, width):
    """
    Initialize a matrix of dimension length*length with zeros and
    an horizontal "band" of iridophores in the matrix center
    Parameters
    -----
    length : int
        Number of lines/columns of the matrix
    width : int
        The width of the iridophores band.

    Returns
    -----
    array
        A matrix which represents the zebrafish lattice
    int
        First line of iridophores band
    int
        Last line of iridophores band
    """
    mat = np.zeros((length,length), dtype=int)
    irid_start = (length//2) - (width//2)
    irid_end = irid_start + width
    return(mat,irid_start,irid_end)

def random_neighbour_of(row,col,length,mat):
    """
    Returns a random neighbor of the cell mat[row,col]
    Parameters
    -----
    row : int
        Current cell row
    col : int
        Current cell column
    length : int
        Number of lines and columns of the matrix.
    mat : array
        Matrix which represents the lattice.

    Returns
    -----
    int
        The value of one of the 8 neighbors
    """
    vertical = random.choice([(row-1)%length,(row+1)%length,row])
    #if the neighbour row is the same as the cell, we cannot chose the same column as the cell
```



```

    if vertical == row:
        horizontal = random.choice([(col-1)%length,(col+1)%length])
    else:
        horizontal = random.choice([(col-1)%length,(col+1)%length,col])
    return (mat[vertical,horizontal])

def display(mat,irid_start,irid_end,h,lx):
    """
    Prints the matrix with colors instead of numbers.
    Empty cell : 0 -> white
    Xantophore : 1 -> yellow
    Melanophore : 2 -> black
    """
    plt.figure()
    plt.pcolormesh(mat,cmap=colors.ListedColormap(['white','yellow','black']))
    if irid_end != 0:
        plt.axhline(y=irid_start,color='red')
        plt.axhline(y=irid_end,color='red')
    plt.title(f" h = {h} and lx = {lx}")
    plt.show()

##### Event functions: each function is related to one of the seven events

def birth_xantophore(cell) :
    """
    First, checks if the cell is empty. Then, if it is the case, a xantophore will be born in this cell.
    -----
    cell : int
        The value of the current cell

    Returns
    -----
    int
        1 if a xantophore is born/if the cell was already occupied by a xantophore
        2 if the cell was already occupied by a melanophore
    """
    if cell == 0:
        return 1
    return cell

def birth_melanophore(row,col,cell,irid_start,irid_end) :
    """
    First, checks if the cell is empty, and is not located on top of the iridophores band.
    Then, if it respects the conditions, a melanophore will be born in this cell.
    Parameters
    -----
    row : int
        Current cell row
    col : int
        Current cell column
    cell : int
        The value of the current cell
    irid_start : int
        First line of iridophores band.
    irid_end : int
        Last line of iridophores band.
    """

```

```

Returns
-----
int
    2 if a melanophore is born/if the cell was already occupied by a melanophore
    1 if the cell was already occupied by a xantophore
    0 if the cell is on top of the iridophores band
'''
if cell == 0 and (row < irid_start or row > irid_end):
    return 2
return cell

def death_xantophore(cell):
    '''
    If the cell is a xantophore, it dies
    -----
    cell : int
        The value of the current cell

    Returns
    -----
    int
        0 if a xantophore is dead
        2 if the cell was occupied by a melanophore
    '''
    if cell == 1:
        return 0
    return cell

def death_melanophore(cell):
    '''
    If the cell is a melanophore, it dies
    -----
    cell : int
        The value of the current cell

    Returns
    -----
    int
        0 if a melanophore is dead
        1 if the cell was occupied by a xantophore
    '''
    if cell == 2:
        return 0
    return cell

def short_range_competitivity_death_xanto(row,col,mat,cell,length) :
    '''
    Fifth event of the algorithm, its represents melanophores inhibition effect on xantophores.
    If the cell is a xantophore, and one the random neighbour is a melanophore, the xantophore dies.
    For borders of the matrix, a neighbour can be the cell on the opposite border.
    Parameters
    -----
    row : int
        Current cell row
    col : int
        Current cell column
    mat : TYPE

```

```

    Matrix which represents the lattice.
    cell : int
    The value of the current cell
    length : int
    Number of lines and columns of the matrix.

Returns
-----
int
    0 if a xantophore is dead
    1 if the neighbour of the xantophore was not a melanophore
    2 if the cell was occupied by a melanophore
'''
if cell == 1:
    neighbour = random_neighbour_of(row,col,length,mat)
    if neighbour == 2:
        return 0
return cell

def short_range_competitivity_death_melano(row,col,mat,cell,length) :
    '''
    Fifth event of the algorithm, its represents xantophores inhibition effect on melanophores.
    If the cell is a melanophore, and one the random neighbour is a xantophore, the melanophore dies.
    For borders of the matrix, a neighbour can be the cell on the opposite border.

    Parameters
    -----
    row : int
    Current cell row
    col : int
    Current cell column
    mat : array
    Matrix which represents the lattice.
    cell : int
    The value of the current cell
    length : int
    Number of lines and columns of the matrix.

    Returns
    -----
    int
    0 if a melanophore is dead
    1 if the neighbour of the melanophore was not a xantophore
    2 if the cell was occupied by a xantophore

    '''
    if cell == 2:
        neighbour = random_neighbour_of(row,col,length,mat)
        if neighbour == 1:
            return 0
    return cell

def long_range_effect_birch_melano(row,col,mat,cell,h,length,irid_start,irid_end) :
    '''
    An random neighbour at a distance h from the current cell is selected.
    If this neighbour is a xantophore and the current cell is empty,

```

```

a melanophore will be born on this cell

Parameters
-----
row : int
    Current cell row
col : int
    Current cell column
mat : array
    Matrix which represents the lattice.
cell : int
    The value of the current cell
h : int
    The distance between the xantophore and its neighbour
length : int
    Number of lines and columns of the matrix.
irid_start : int
    First line of iridophores band.
irid_end : int
    Last line of iridophores band.

Returns
-----
int
    2 if a melanophore is born/if the cell was already occupied by a melanophore
    1 if the cell was already occupied by a xantophore
    0 if the cell is the selected neighbour was not a xantophore

'''
# obtenir l'angle teta entre 0 et 2pi
teta = random.uniform(0, (2 * math.pi))
# les coordonnées du voisin à distance h (on en choisit qu'un seul)
delta_x = round(h * math.cos(teta))
delta_y = round(h * math.sin(teta))
if cell == 0 and (row < irid_start or row > irid_end):
    neighbour = mat[(row+delta_y)%length,(col+delta_x)%length]
    if neighbour == 1:
        return 2
return cell

def run(lx,h,irid):
    '''
    Initializes a matrix which represents zebrafish lattice, eventually with an iridophores band
    Simulates the process according to parameters. Multiple iterations (default = 10) are made.
    For each one, a random cell is selected,
    and a random event occurs (according to probabilities given in the article).
    At the end, the matrix is converted to a plot, which represents the lattice with colors.

    Parameters
    -----
    lx : int
        Strength of enhancement effect of xantophores on melanophores
    h : int
        The required distance for melanophore to enhance xantophores birth
    irid : int
        The width of the iridophores band.

    '''

```

```

bx = 1 # birth rate
dx = 0 # death rate
sx = 1 # xantophores strength

bm = 0 # birth rate
dm = 0 # death rate
sm = 1 # melanophores strength

length = 100 # number of lines/columns of the matrix
nb_iter = 1000000000 #number of iterations

# Matrix initialization
if irid == 0:
    mat = np.zeros((length,length),dtype=int)
    #if there is not iridophores band, we define irid_start and irid_end, as the boundaries of the matrix
    irid_start = length
    irid_end = 0

else:
    (mat,irid_start,irid_end) = initialization_with_iridophores(length, irid)

# Events rates
sum_rates = bx + bm + dx + dm + sm + sx + lx
pBX = bx/sum_rates
pBM = bm/sum_rates
pDX = dx/sum_rates
pDM = dm/sum_rates
pCDX = sm/sum_rates
pCDM = sx/sum_rates

for time in range (nb_iter) :
    row = random.randint(0, length-1)
    col = random.randint(0, length-1)
    cell = mat[row,col]
    Pk = random.random() #probabilité de l'événement pour cette itération

    if Pk <= pBM :
        mat[row,col] = birth_melanophore(row,col,cell,irid_start,irid_end)

    elif Pk <= pBX + pBM :
        mat[row,col] = birth_xantophore(cell)

    elif Pk <= pBX + pBM + pDX :
        mat[row,col] = death_xantophore(cell)

    elif Pk <= pBX + pBM + pDX + pDM :
        mat[row,col] = death_melanophore(cell)

    elif Pk <= pBX + pBM + pDX + pDM + pCDX :
        mat[row,col] = short_range_competitivy_death_xanto(row,col,mat,cell,length)

    elif Pk <= pBX + pBM + pDX + pDM + pCDX + pCDM :
        mat[row,col] = short_range_competitivy_death_melano(row,col,mat,cell,length)

    else:
        mat[row,col] = long_range_effect_birth_melano(row,col,mat,cell, h, length,irid_start,irid_end)

    if time%100000000 == 0:

```

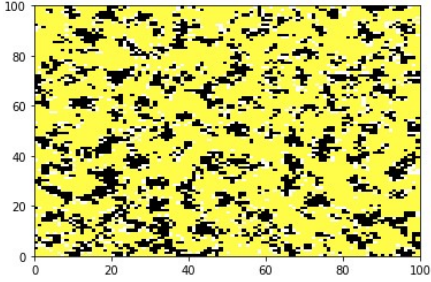
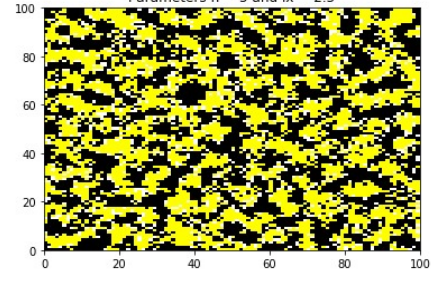
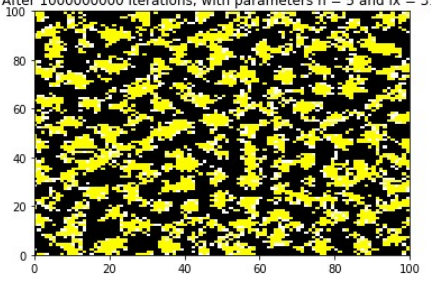
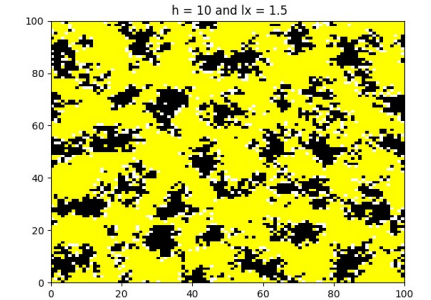


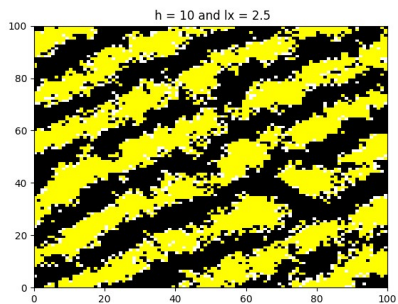
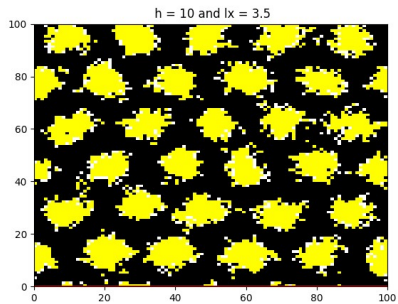
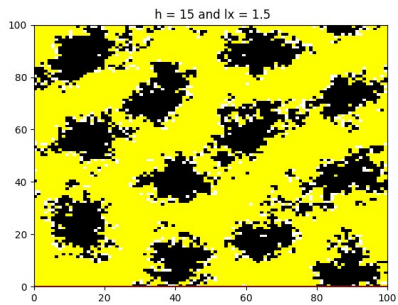
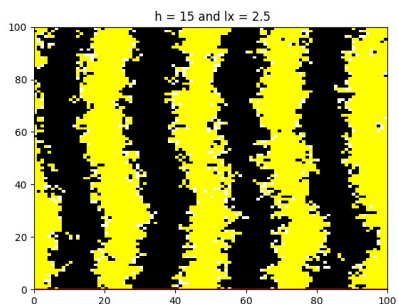
```
print(f"{time} iterations")

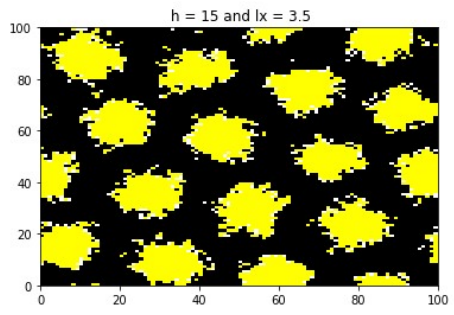
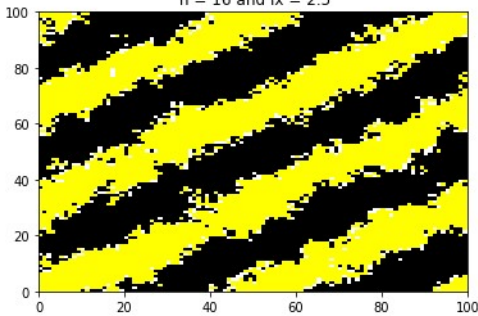
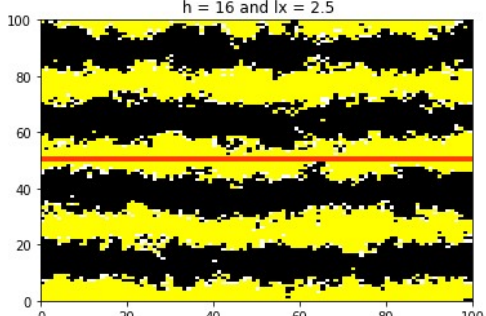
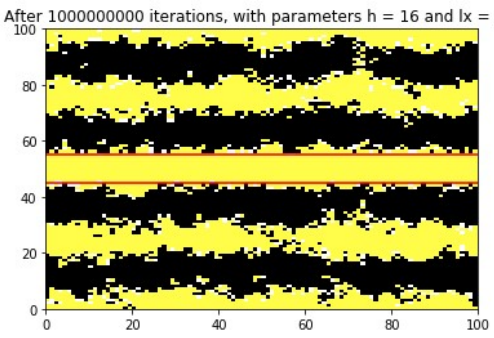
display(mat, irid_start, irid_end, h, lx)

#Function calling
run(lx=2.5, h=16, irid=1)
```

Annexe 2 – Motifs obtenus pour différentes valeurs des paramètres h et lx

Paramètres		Motif
Pas d'iridophores	$h=5, lx=1.5$	<p>h = 5 and lx = 1.5</p> 
	$h=5, lx=2.5$	<p>Parameters h = 5 and lx = 2.5</p> 
	$h=5, lx=3.5$	<p>After 1000000000 iterations, with parameters h = 5 and lx = 3.5</p> 
	$h=10, lx=1.5$	<p>h = 10 and lx = 1.5</p> 

	$h=10, lx=2.5$	 <p>h = 10 and lx = 2.5</p>
	$h=10, lx=3.5$	 <p>h = 10 and lx = 3.5</p>
	$h=15, lx=1.5$	 <p>h = 15 and lx = 1.5</p>
	$h=15, lx=2.5$	 <p>h = 15 and lx = 2.5</p>

	$h=15, lx=3.5$	 <p>h = 15 and lx = 3.5</p>
	$h=16, lx=2.5$	 <p>h = 16 and lx = 2.5</p>
Les iridophores représentent une bande de la matrice	$h=16, lx=2.5$	 <p>h = 16 and lx = 2.5</p>
Les iridophores représentent dix bandes de la matrice	$h=16, lx=2.5$	 <p>After 10000000000 iterations, with parameters h = 16 and lx = 2.5</p>