Processus de réaction-diffusion

« Le(s) modèle(s) de Alan Turing »

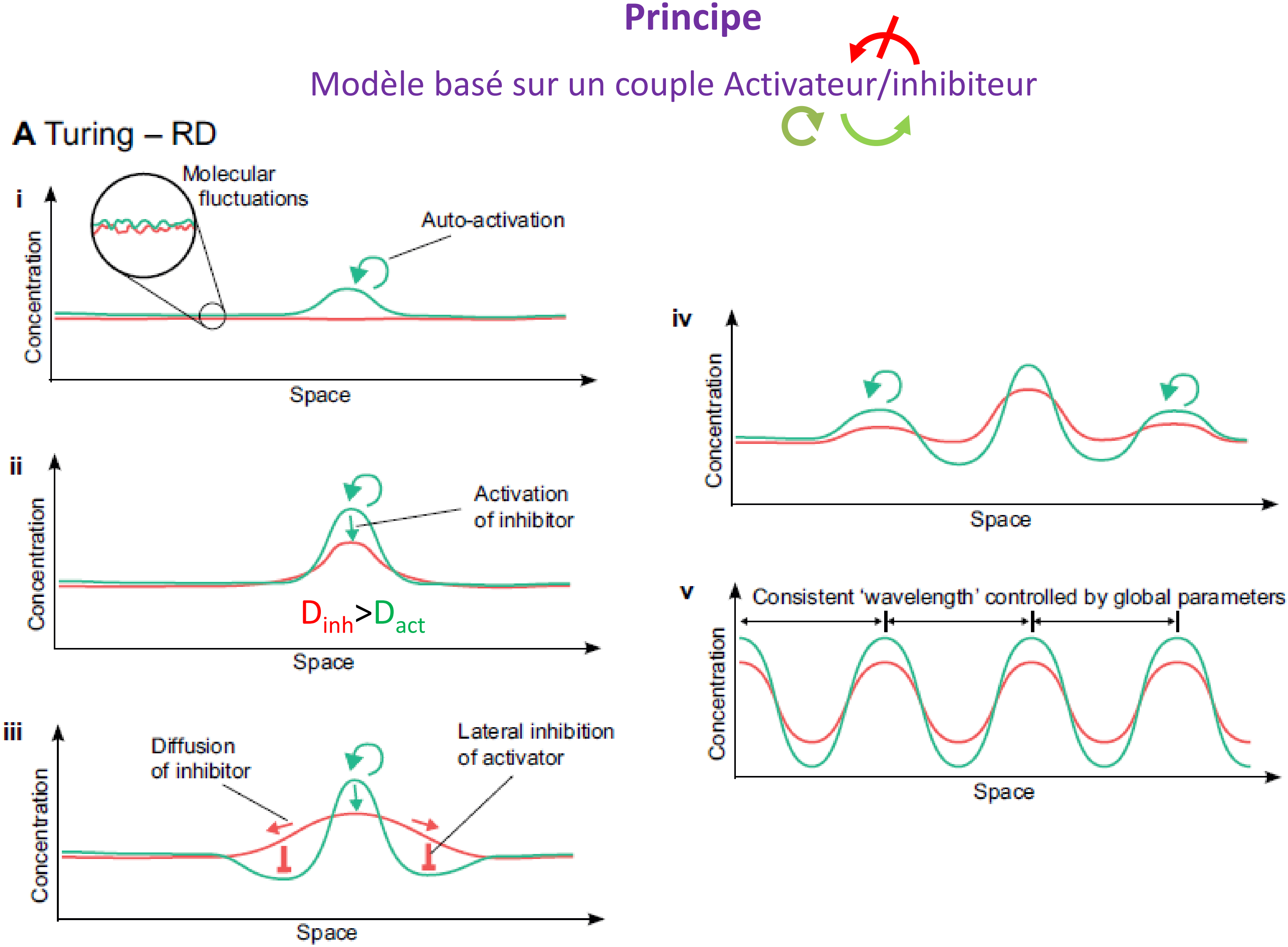
(1912 – 1954)

Emergence spontanée de motifs réguliers ou de phénomènes cycliques (Rythme circadien, horloge biologique …)

H. Meinhardt, The Algorithmic Beauty of Seashells (Springer-Verlag, Berlin, 1995).

Principe

Modèle basé sur un couple Activateur/inhibiteur



Jeremy B. A. Green and James Sharpe, *Positional information and reaction-diffusion: two big ideas in developmental biology combine*, Development **142**, 1203-1211 (2015) doi:10.1242/dev.114991

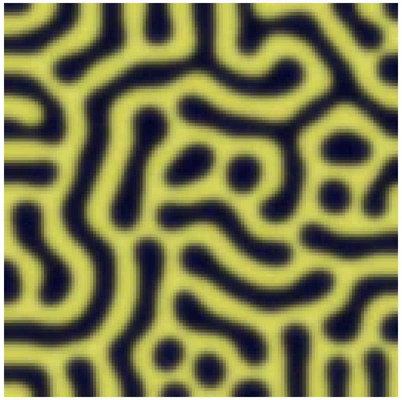
Equations

Modèle basé sur un couple Activateur/inhibiteur

Evolution temporelle des concentrations spatiales en espèces u (activateur) et v (inhibiteur)

$$\left[\begin{aligned} \frac{\partial \mathrm{u}}{\partial \mathrm{t}} &= \mathrm{F}(\mathrm{u},\mathrm{v}) + \mathrm{D_u}\Delta \mathrm{u} \\ \frac{\partial \mathrm{v}}{\partial \mathrm{t}} &= \mathrm{G}(\mathrm{u},\mathrm{v}) + \mathrm{D_v}\Delta \mathrm{v} \end{aligned}\right]$$

Equations locales



$D_v \gg D_u$
$b_u < 0$

Production / Dégradation

Diffusion

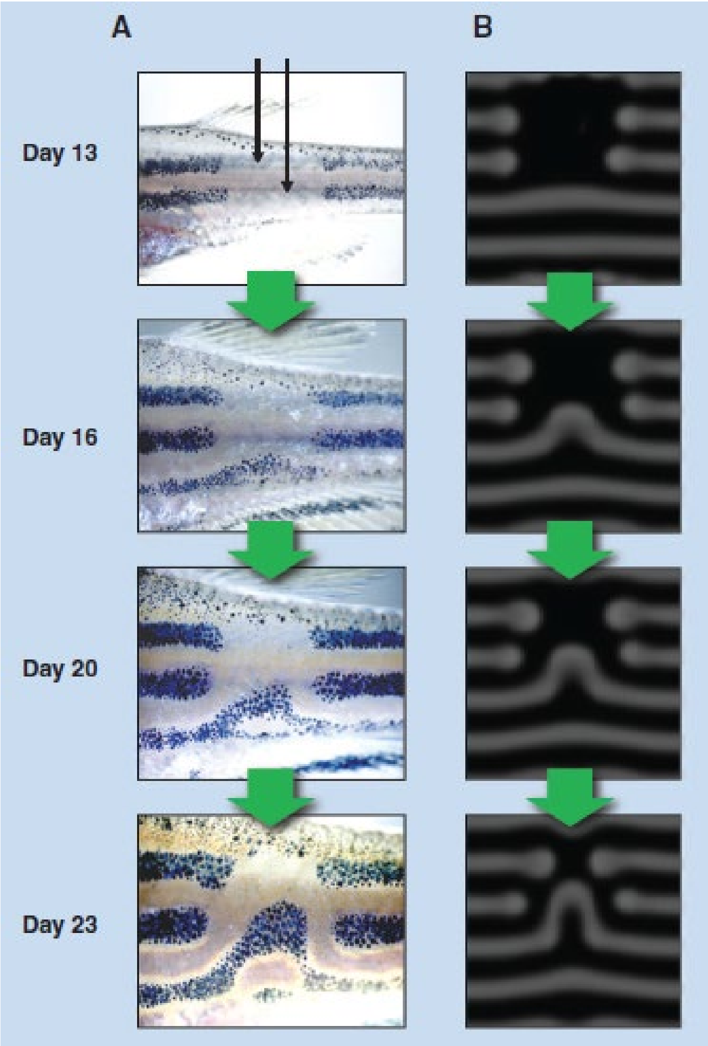
un choix possible…

$$\boxed{\begin{aligned} \mathrm{F}(\mathrm{u},\mathrm{v}) &= \mathrm{F_c}(\mathrm{u},\mathrm{v}) - \alpha_u \mathrm{u} = (a_u u + b_u v + c_u) - \alpha_u \mathrm{u} \\ \mathrm{G}(\mathrm{u},\mathrm{v}) &= \mathrm{G_c}(\mathrm{u},\mathrm{v}) - \alpha_v \mathrm{v} = (a_v u + b_v v + c_v) - \alpha_v \mathrm{v} \end{aligned}}$$

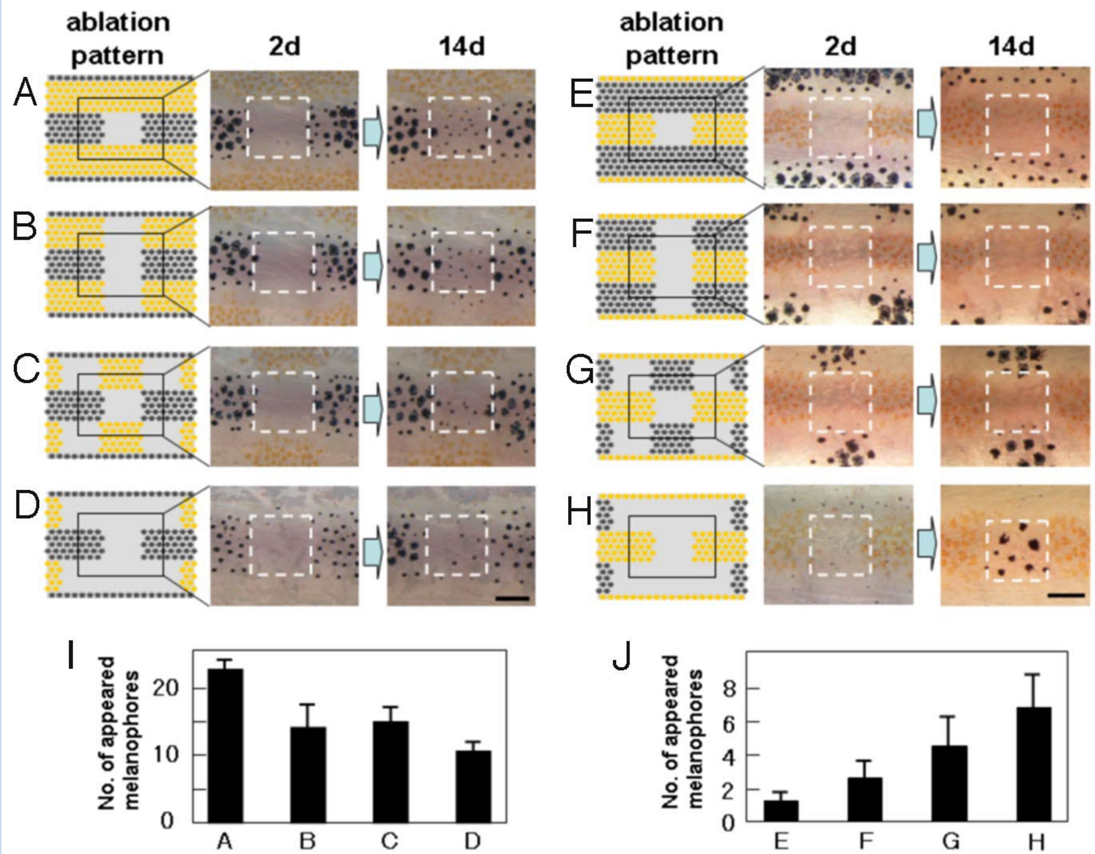
Shigeru Kondo and Takashi Miura, *Reaction-Diffusion Model as a Framework for Understanding Biological Pattern Formation*, Science **329**, 1616-1620 (2010),
DOI: 10.1126/science.1179047

Exemple du Poisson-zèbre

Nombre de jours après dépigmentation



Modélisation



Pigment noir : Cellules *melanophores*

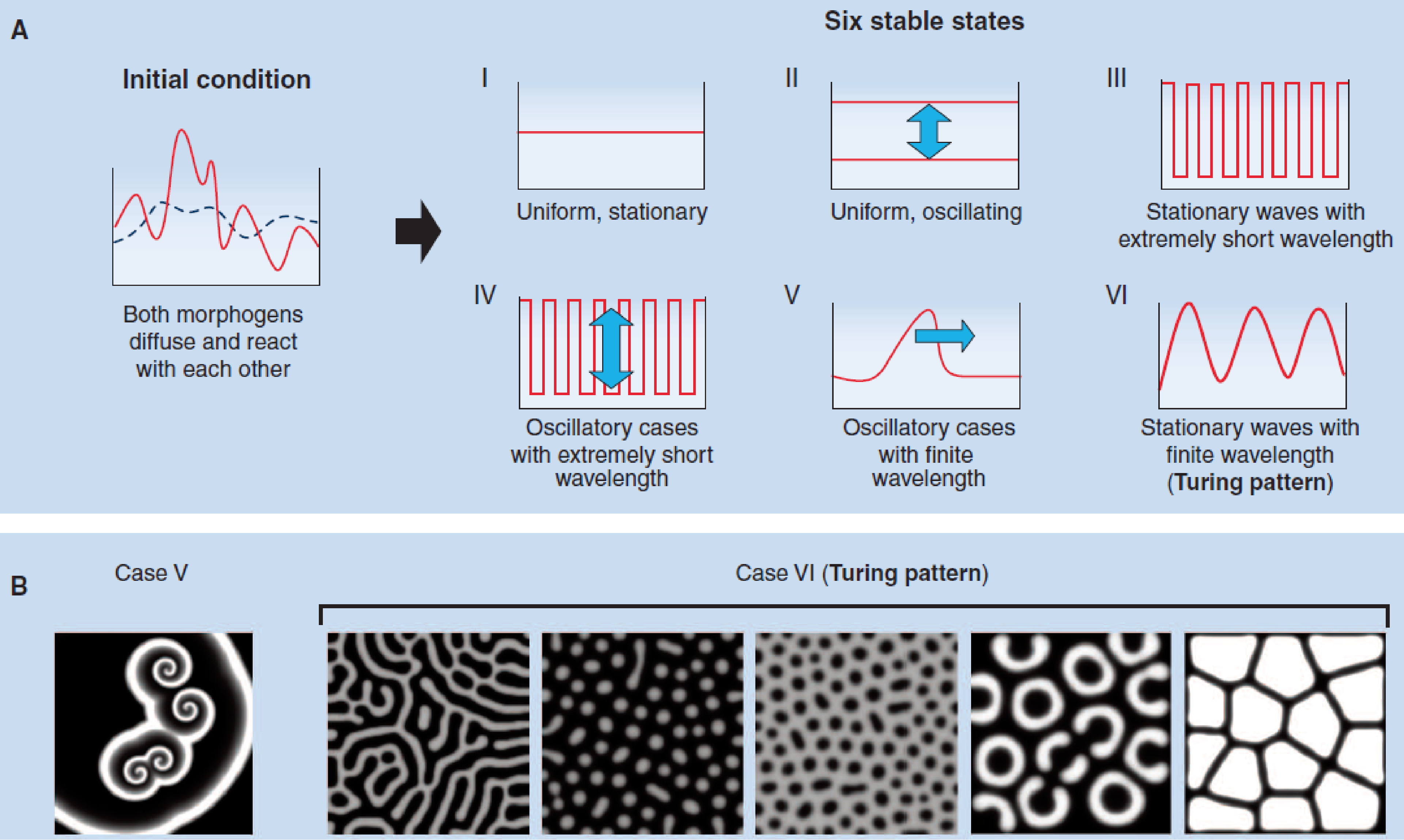
Pigment jaune: Cellules *xanthophores*

Shigeru Kondo and Takashi Miura, *Reaction-Diffusion Model as a Framework for Understanding Biological Pattern Formation*, Science **329**, 1616-1620 (2010), DOI: 10.1126/science.1179047

A. Nakamasu, G. Takahashi, A. Kanbe, S. Kondo, Interactions between zebrafish pigment cells responsible for the generation of Turing patterns, PNAS **106**, 8429 (2009).

Etats stables

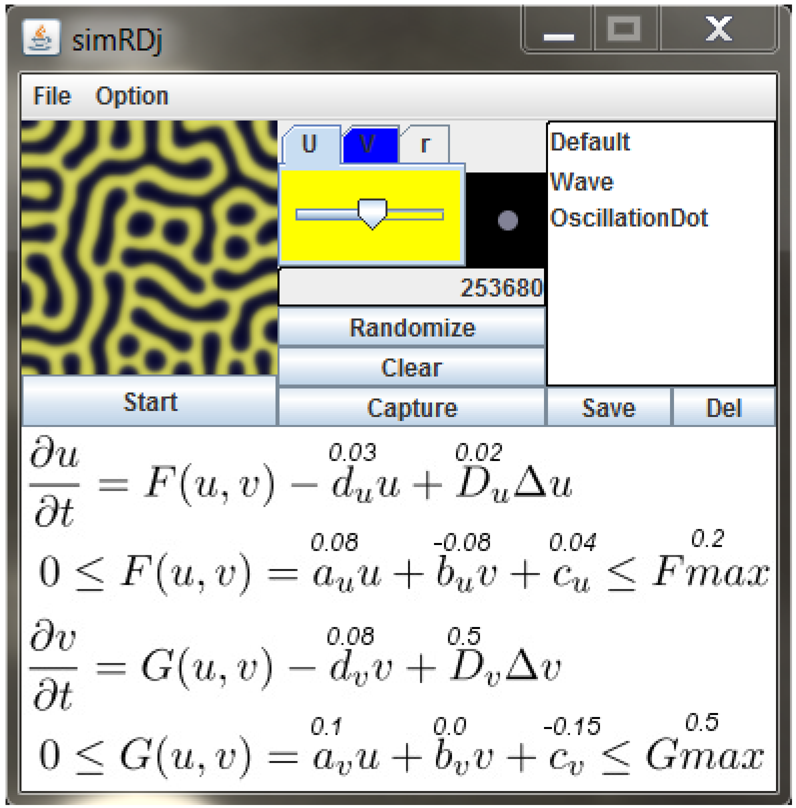
Modèle basé sur un couple Activateur/inhibiteur



Shigeru Kondo and Takashi Miura, *Reaction-Diffusion Model as a Framework for Understanding Biological Pattern Formation*, Science **329**, 1616-1620 (2010), DOI: 10.1126/science.1179047

Modélisation

Programme fourni en *supplementary information* de Kondo *et al.* (Mais java…)



Shigeru Kondo and Takashi Miura, *Reaction-Diffusion Model as a Framework for Understanding Biological Pattern Formation*, Science **329**, 1616-1620 (2010), DOI: 10.1126/science.1179047
https://science.sciencemag.org/content/suppl/2010/09/22/329.5999.1616.DC1

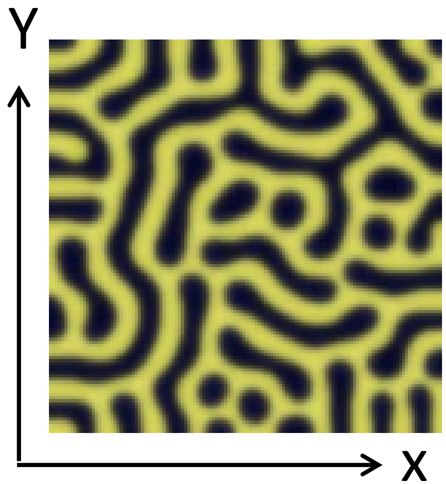
Projet sous R

$$\frac{\partial u}{\partial t} = F(u,v) + D_u \Delta u$$
$$\frac{\partial v}{\partial t} = G(u,v) + D_v \Delta v$$

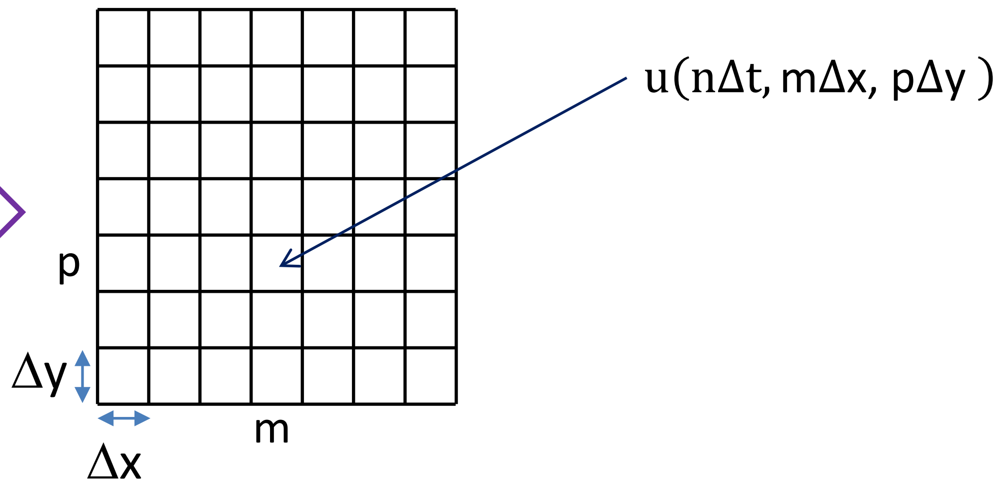
Différences finies : $t = n\,\Delta t$

$$u(t+\Delta t) = u(t) + \Delta t * (\ F(u(t),v(t)) + D_u \Delta u(t)\ )$$
$$v(t+\Delta t) = v(t) + \Delta t * (\ G\big(u(t),v(t)\big) + D_v \Delta v(t)\ )$$

<u>Discrétisation de l’espace: Deux tableaux, un pour u et un pour v</u>



“Color map”, ici distribution de l’espèce u

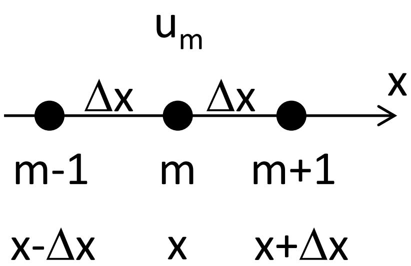


(m,p)

$X = m\,\Delta x$

$Y = p\,\Delta y$

Projet sous R



Développement de Taylor

$$\begin{cases} u_{m+1} = u_m + \Delta x \left(\dfrac{\partial u}{\partial x}\right)_m + \dfrac{\Delta x^2}{2}\left(\dfrac{\partial^2 u}{\partial x^2}\right)_m + \mathrm{O}\left(\Delta x^3\right) \\ u_{m-1} = u_m - \Delta x \left(\dfrac{\partial u}{\partial x}\right)_m + \dfrac{\Delta x^2}{2}\left(\dfrac{\partial^2 u}{\partial x^2}\right)_m + O\left(\Delta x^3\right) \end{cases}$$

Dérivée première (1D)

$$u_{m+1} - u_{m-1} = 2\Delta x \left(\frac{\partial u}{\partial x}\right)_m + O\left(\Delta x^3\right)$$

D'où

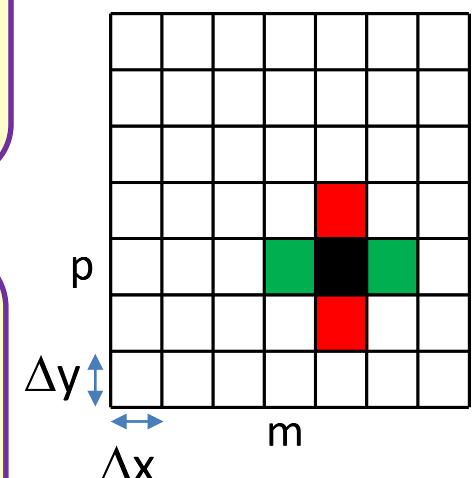
$$\boxed{\left(\frac{\partial u}{\partial x}\right)_m = \frac{u_{m+1} - u_{m-1}}{2\Delta x} + O\left(\Delta x^2\right)}$$

Dérivée seconde (2D)

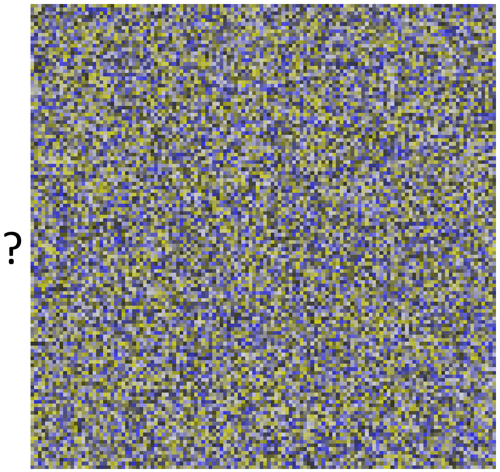
$$u_{m+1} + u_{m-1} = 2u_m + \Delta x^2 \left(\frac{\partial^2 u}{\partial x^2}\right)_m + O\left(\Delta x^4\right)$$

D'où

$$\boxed{\left(\frac{\partial^2 u}{\partial x^2}\right)_m = \frac{u_{m+1} - 2u_m + u_{m-1}}{\Delta x^2} + O\left(\Delta x^2\right)}$$



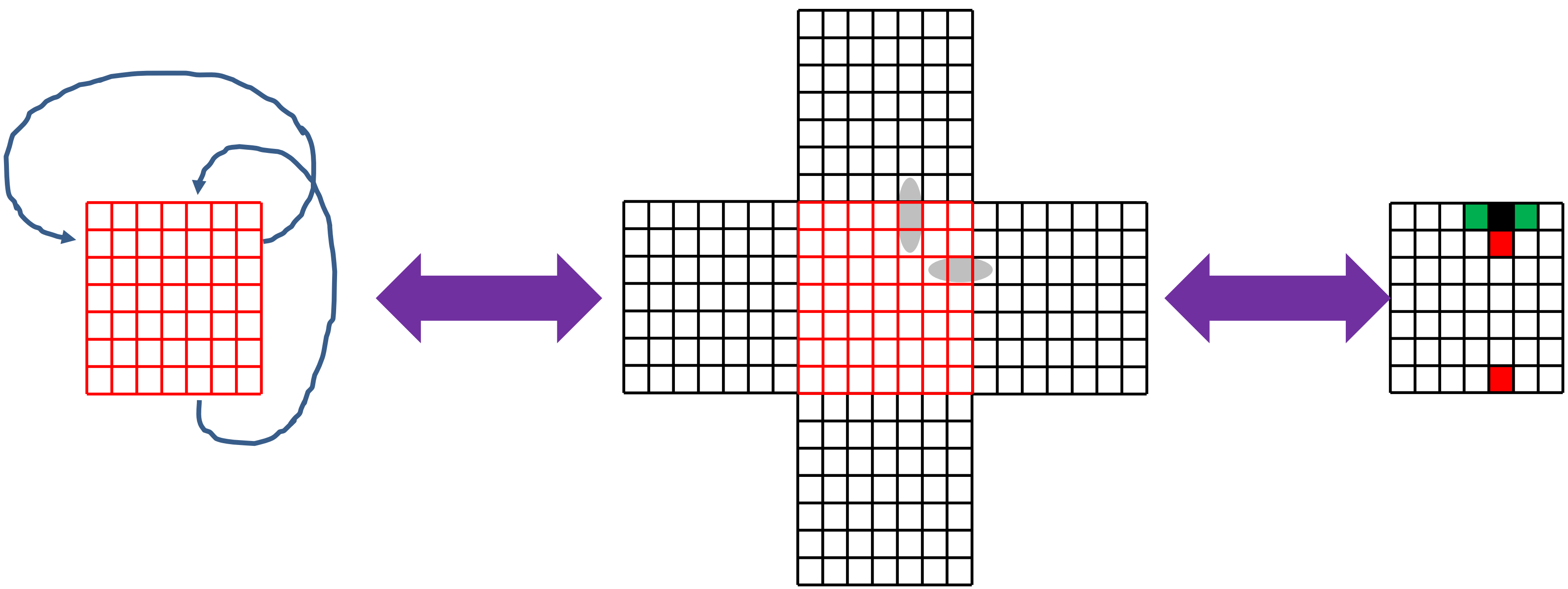
Projet sous R



Conditions initiales : On tire des valeurs aléatoires pour u et v

… sur quelle gamme de concentration ?

Conditions aux limites : conditions périodiques



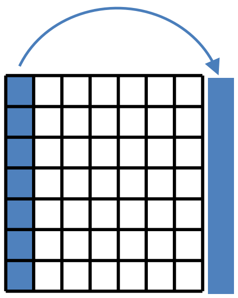
Projet sous R

Gestion des matrices :
https://fr.wikibooks.org/wiki/Programmer_en_R/Manipuler_les_matrices

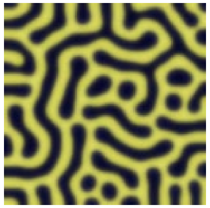
Créer une matrice :
- dim<-10
- M <- matrix(data=rep(0,dim*dim),nrow=dim,ncol=dim)   <- contient des zéros
- P <- matrix(data=runif(dim*dim),nrow=dim,ncol=dim))   <- contient des nombres aléatoires

Accès aux éléments:
- M[i, j] désigne l'élément de la ligne i et de la colonne j de la matrice M
- M[, j] permet d'extraire la colonne j de la matrice M
- M[i,] permet d'extraire la ligne i de la matrice M

Combiner
- cbind(M[,2:dim],M[,1]) : manipulation des colonnes
- rbind(M [2:dim,],M [1,]) : manipulation des lignes



Affichage des valeurs d'une matrice :
- filled.contour(z = M, color = terrain.colors)
- filled.contour(z = M, nlevels = 100, color = terrain.colors)



Manipuler les éléments, exemple:
Fuv=Fuv*(Fuv>=0); #condition Fuv>=0

Matrices :

Mu, Mv : Matrices contenant les concentrations en activateurs et inhibiteurs

Fuv et Guv : Matrice permettant de calculer les termes de Production / Dégradation

Diffu, Diffv : Matrice permettant de calculer les termes de diffusion

$$u(t+\Delta t) = u(t) + \Delta t * (\ F(u(t), v(t)) + D_u \Delta u(t)\ )$$

$$v(t+\Delta t) = v(t) + \Delta t * (\ G\big(u(t), v(t)\big) + D_v \Delta v(t)\ )$$

Sauvegarder une image :

```
ipng=0

for (i in seq(1, Niter+1)) #Boucle de Niter itérations du pas temporel
{

  if(i%%100 == 1) #Sauvegarde d'une image sur 100 pour la matrice Mu
(concentration activateur)
  {
    out=paste0("out",toString(ipng),".png")
    png(file = out, width = 800, height = 800)
    filled.contour(z = Mu, nlevels = 100, color = terrain.colors)
    dev.off()
    ipng=ipng+1
  }

....
}
```