Intégration d'équations différentielles avec Python

Camille Chambon

INRA – EcoSys

Kfé Sciences du 17 juin 2016

Plan

1. **Introduction sur les équations différentielles**
2. **Résolution numérique d'équations différentielles avec Python : la bibliothèque logicielle « SciPy.Integrate »**
3. **Application à la modélisation biologique du blé : le modèle à compartiments « CN-Wheat »**
4. **Pour se détendre...**

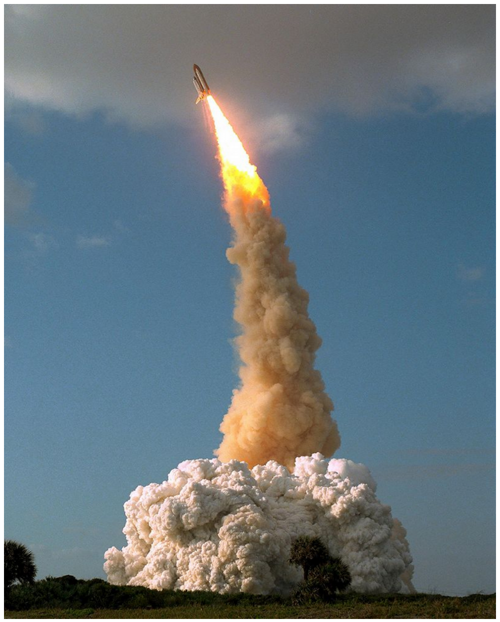
1. Introduction sur les équations différentielles

- **C'est quoi une équation différentielle ?**
  - **Relation** entre une ou plusieurs **fonctions** inconnues et leurs **dérivées**.
  - <u>Exemple</u> : $f'(x)=3\cdot f(x)-5$
    - $f(x)$: fonction inconnue,
    - $f'(x)$: sa dérivée.

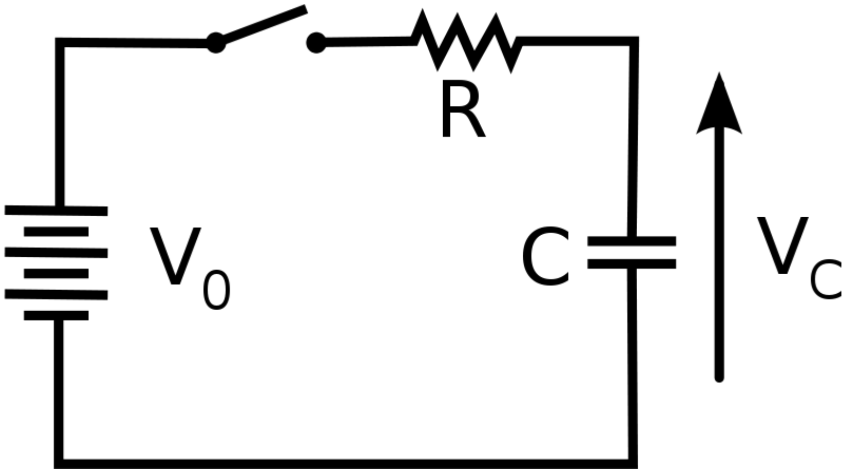
1. Introduction sur les équations différentielles

- **Ça sert à quoi ?**
  - Traduire, sous forme de **modèles** mathématiques, les lois qui régissent la **variation** de telle ou telle **grandeur**.
  - <u>Exemple</u> : position d'une navette spatiale, charge d'un condensateur électrique, concentration d'un produit lors d'une réaction chimique, effectif d'une population, etc.

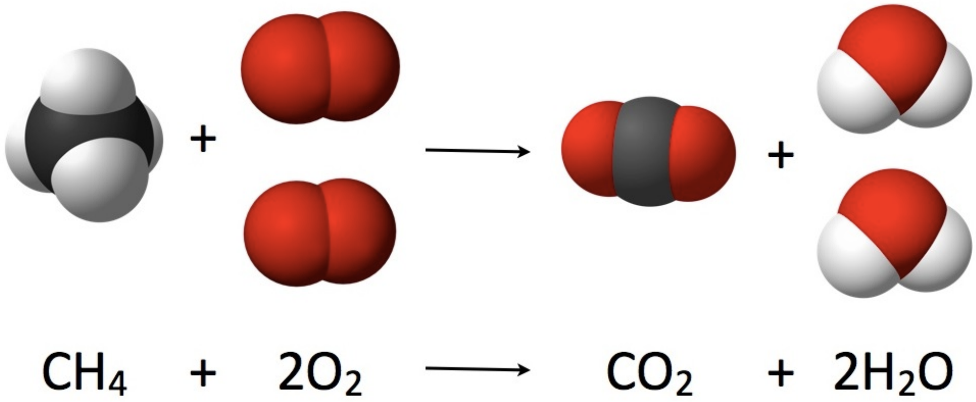
1. Introduction sur les équations différentielles



Décollage de la navette Discovery

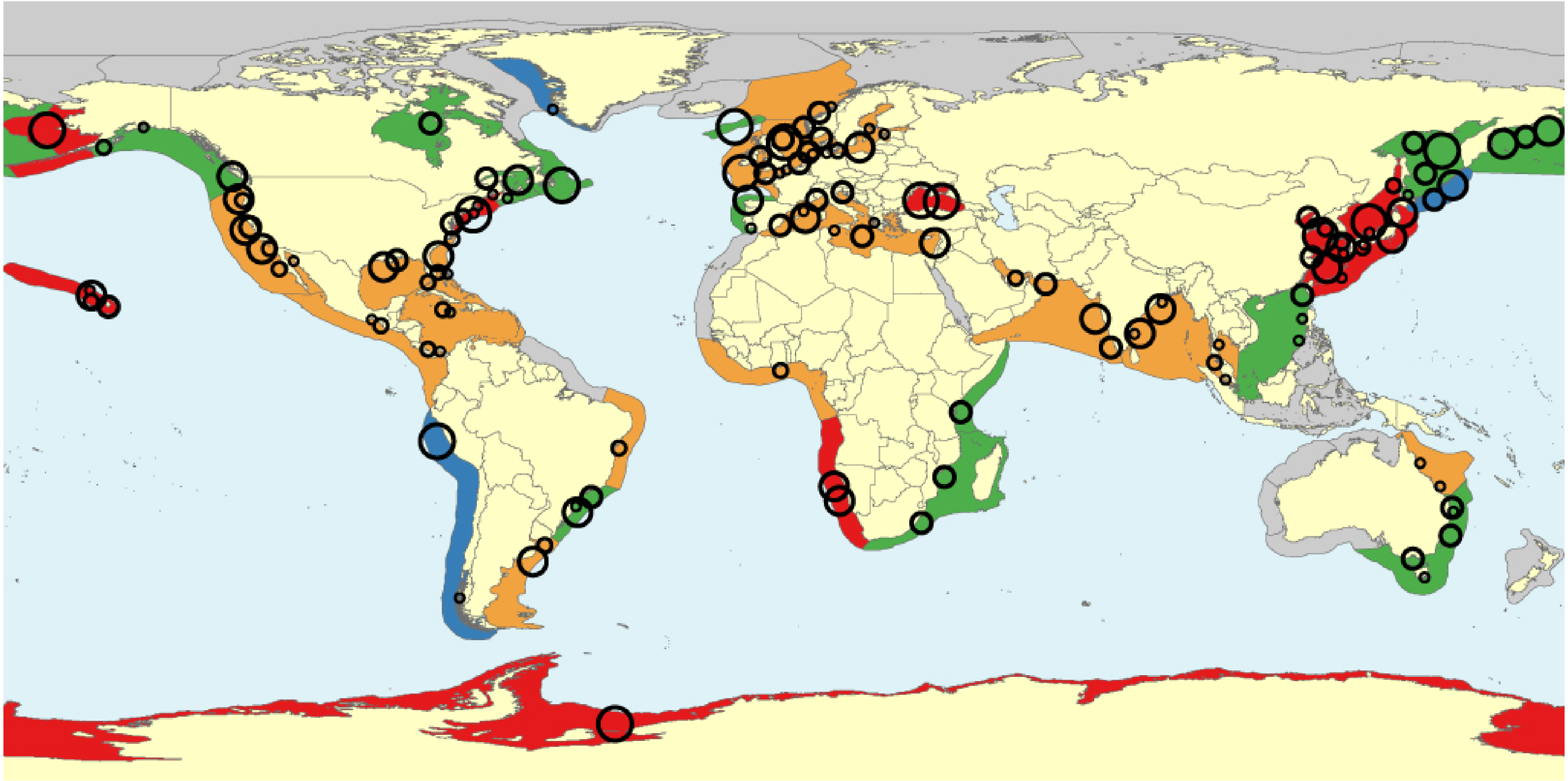


Charge d'un condensateur



Réaction de combustion du méthane

1. Introduction sur les équations différentielles



Evolution des populations de méduses

1. Introduction sur les équations différentielles

- **Ça veut dire quoi « résoudre une équation différentielle » ?**
  - **Chercher** toutes les **fonctions** vérifiant l'équation différentielle proposée
  - <u>Exemple</u> : résoudre l'équation $f'(x)=3\cdot f(x)-5$ sur l'intervalle I, c'est chercher toutes les fonctions $f(x)$ dérivables sur I et vérifiant pour tout x de I : $f'(x)=3\cdot f(x)-5$

1. Introduction sur les équations différentielles

- **Comment fait-on pour résoudre les équations différentielles « classiques » ?**
  - Équations différentielles dont les solutions peuvent être exprimées au moyen de **fonctions élémentaires**.
  - **Résolution explicite**
  - <u>Exemple</u> : solutions de $f'(x)=3\cdot f(x)-5$ sur $\mathbb{R}$:

    $f(x)=C\cdot e^{3\cdot x}+\frac{5}{3}$, avec C constante.

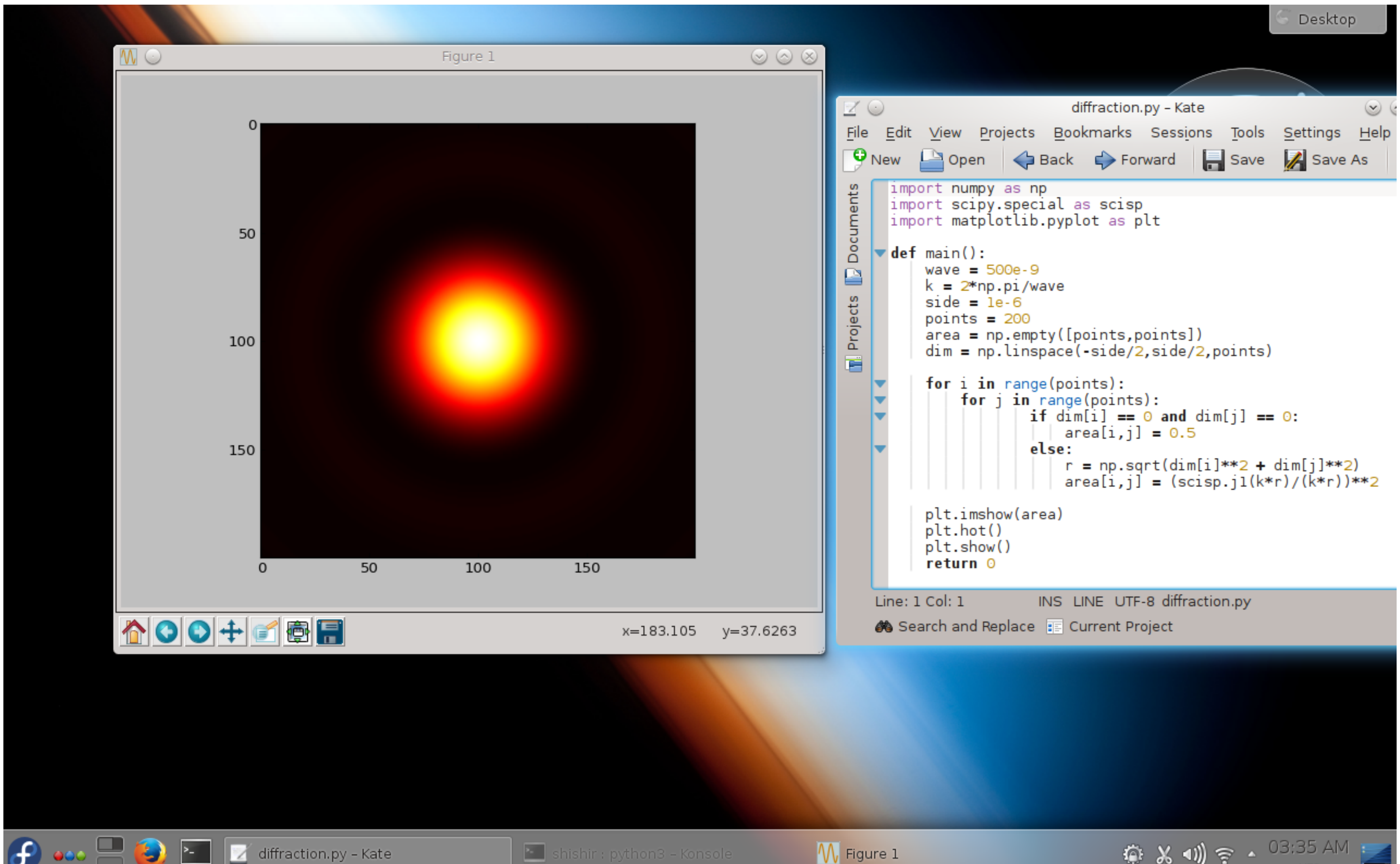
1. Introduction sur les équations différentielles

- **Comment fait-on pour résoudre les autres équations différentielles ?**
  - **Résolution numérique**
  - Utilisation de méthodes permettant d'**approcher** numériquement les solutions.
    - <u>Exemples de méthodes</u> : *Euler*, *Runge-Kutta*, *Newmark*, *différences finies*, *éléments finis*.

2. Résolution numérique d'équations différentielles avec Python : la bibliothèque logicielle « SciPy.Integrate »

- **C'est quoi SciPy ?**
  - Ensemble de **bibliothèques Python** à usage **scientifique**.
  - Environnement de travail similaire à Scilab, GNU Octave, Matlab, R.
  - Optimisation, algèbre linéaire, statistiques, traitement du signal, traitement d'images, etc.
  - **Visualisation graphique** avec matplotlib
  - Codée en **C** et **Fortran**
  - Licence **libre**
  - https://www.scipy.org/

2. Résolution numérique d'équations différentielles avec Python : la bibliothèque logicielle « SciPy.Integrate »



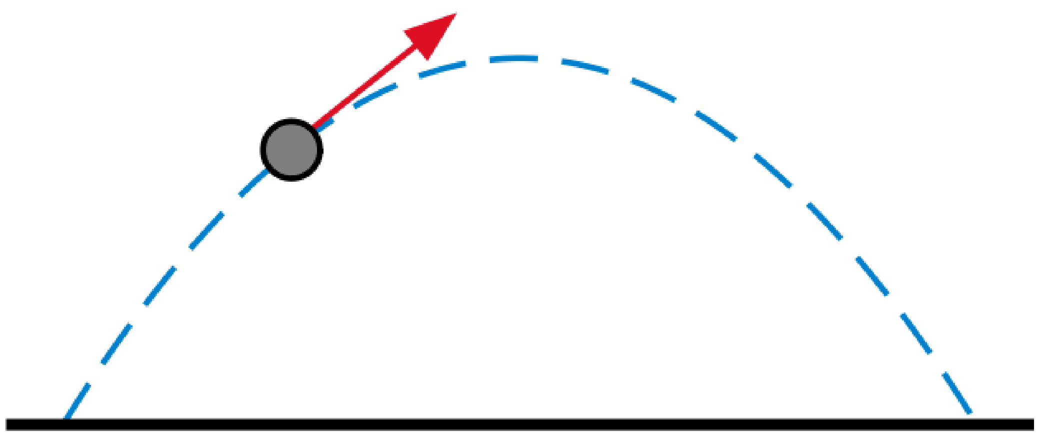
SciPy couplée à Matplotlib (diagramme de diffraction)

2. Résolution numérique d'équations différentielles avec Python : la bibliothèque logicielle « SciPy.Integrate »

- **C'est quoi « SciPy.Integrate » ?**
  - Ensemble de routines pour l'**intégration** numérique de **fonctions** et de systèmes d'**équations différentielles**
  - http://docs.scipy.org/doc/scipy/reference/integrate.html

2. Résolution numérique d'équations différentielles avec Python : la bibliothèque logicielle « SciPy.Integrate »

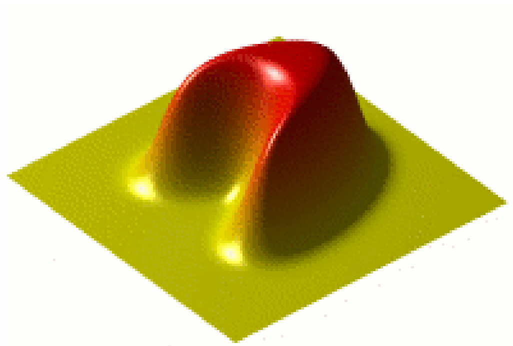
- **Quels types d'équations différentielles peut-on résoudre avec cette bibliothèque ?**
  - Équations différentielles **ordinaires** du **premier ordre**, ou pouvant se ramener à un système d'équations différentielles ordinaires du premier ordre.



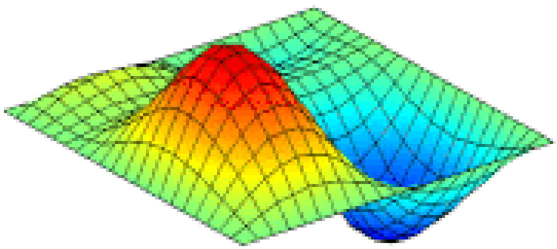
Trajectoire d'un boulet de canon

2. Résolution numérique d'équations différentielles avec Python : la bibliothèque logicielle « SciPy.Integrate »

- **Quels types d'équations différentielles NE peut-on PAS résoudre avec cette bibliothèque ?**
  - Équations aux dérivées partielles (**EDP**).



Conduction thermique



Propagation d'une onde

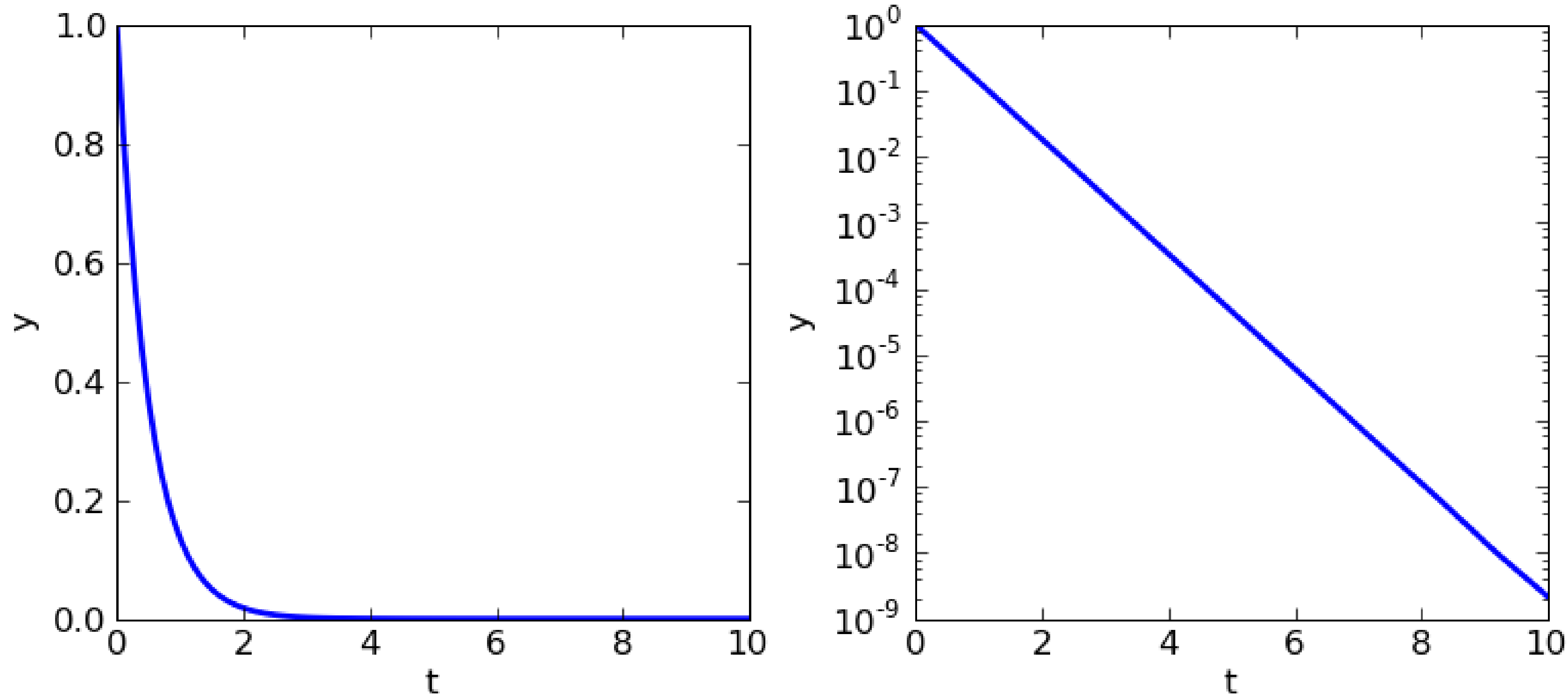
  - Pour les EDP : voir bibliothèque ***fipy*** (http://www.ctcms.nist.gov/fipy/)

2. Résolution numérique d'équations différentielles avec Python : la bibliothèque logicielle « SciPy.Integrate »

- **Comment utiliser la bibliothèque SciPy.Integrate ?**
  - Routine **scipy.integrate.odeint(...)**
  - Exemple : $\frac{dy}{dt} = -2 \cdot y$, avec $t = 0..10$ et $y(t=0) = 1$

```
>>> from scipy.integrate import odeint
>>> def rhs(y, t): # second membre de l'equa diff
...         return -2*y
>>> t = np.linspace(0, 10, 100)
>>> y = odeint(rhs, 1, t) # solution
```

2. Résolution numérique d'équations différentielles avec Python : la bibliothèque logicielle « SciPy.Integrate »

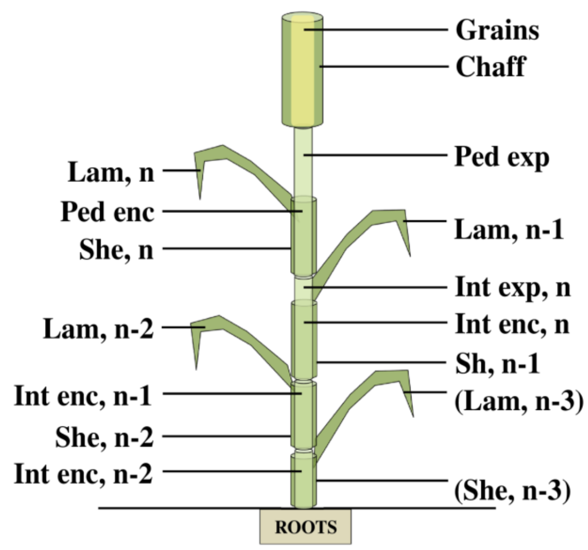


2. Résolution numérique d'équations différentielles avec Python : la bibliothèque logicielle « SciPy.Integrate »

- Routine **scipy.integrate.odeint(...)**
  - Utilise le solveur ***lsoda*** de la bibliothèque Fortran ***odepack***
  - Pour plus d'informations : voir doc en ligne http://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html
- Autre routine plus générique : **scipy.integrate.ode(...)**
  - Voir documentation en ligne : http://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.ode.html

3. Application à la modélisation biologique du blé : le modèle à compartiments « CN-Wheat »

- **Présentation du modèle (d'après R.Barillot, iCROPM2016)**
  - Modèle de Plante Structure-Fonction (**FSPM**)
  - Métabolisme **Carbone-Azote** dans le **blé**
  - Plante <=> **organes interconnectés** : racines, entre-nœuds, gaines, limbes, pédoncules, épis et grains

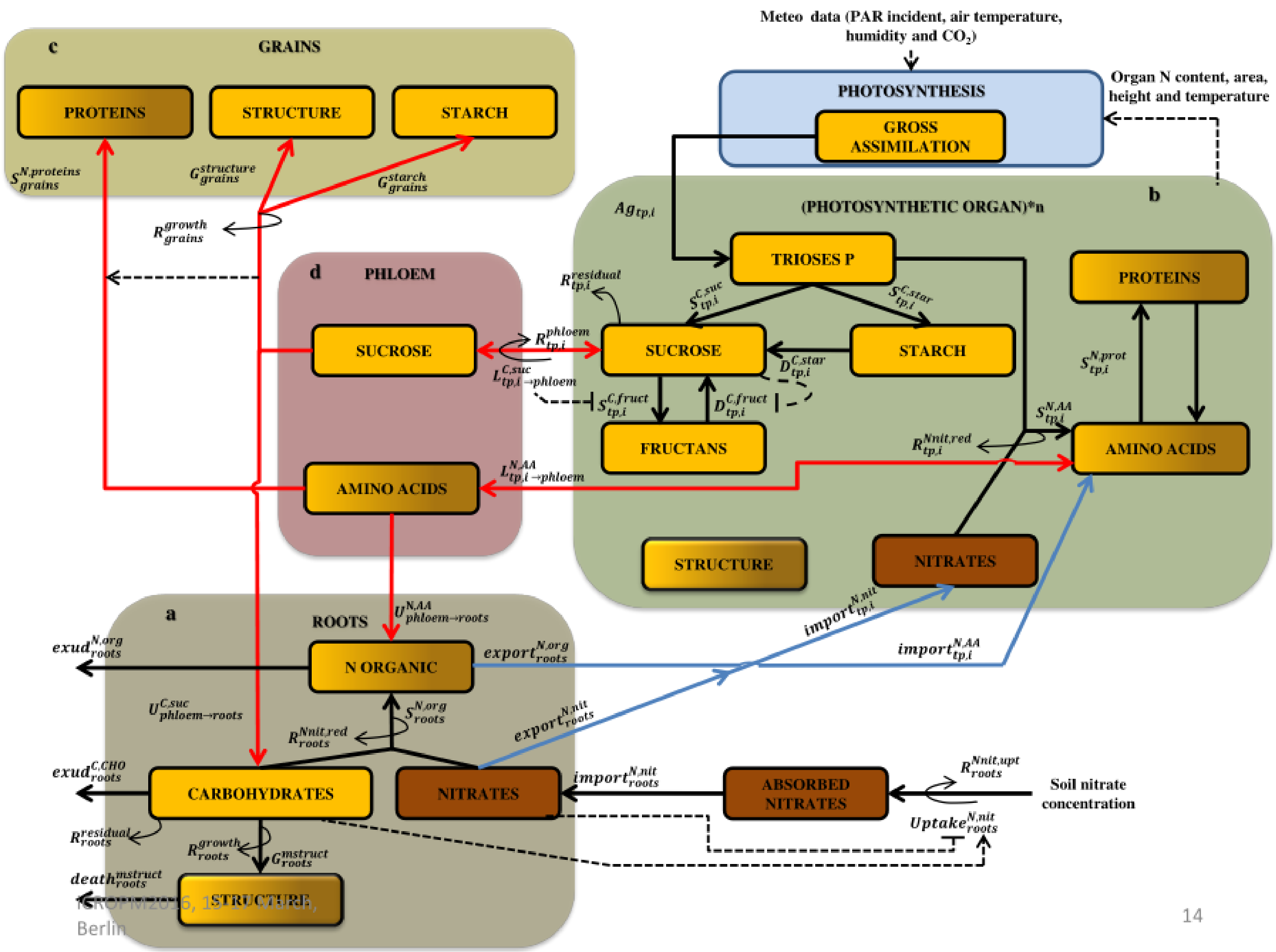


Structure de la plante (R.Barillot, iCROPM2016)

3. Application à la modélisation biologique du blé : le modèle à compartiments « CN-Wheat »

- **Présentation du modèle (suite)**
  - Organe <=> ensemble de **compartiments**
  - Compartiments <=> **concentrations** de **métabolites** : fructanes, amidon, protéines, saccharose, acides aminés, nitrates.
    - Concentrations des métabolites varient en fonction de **processus physiologiques**
    - Processus physiologiques régis par concentrations de métabolites
  - **Interactions** avec **rétroactions** entre les compartiments

3. Application à la modélisation biologique du blé : le modèle à compartiments « CN-Wheat »



Interactions entre les compartiments (R.Barillot, iCROPM2016)

3. Application à la modélisation biologique du blé : le modèle à compartiments « CN-Wheat »

- **Mise en équations du modèle**
  - Interactions décrites par un système d'équations différentielles
  - Calculer les concentrations $c_1, c_2, \ldots, c_m$ à chaque pas de temps t

    <=> résoudre le système d'équations différentielles :

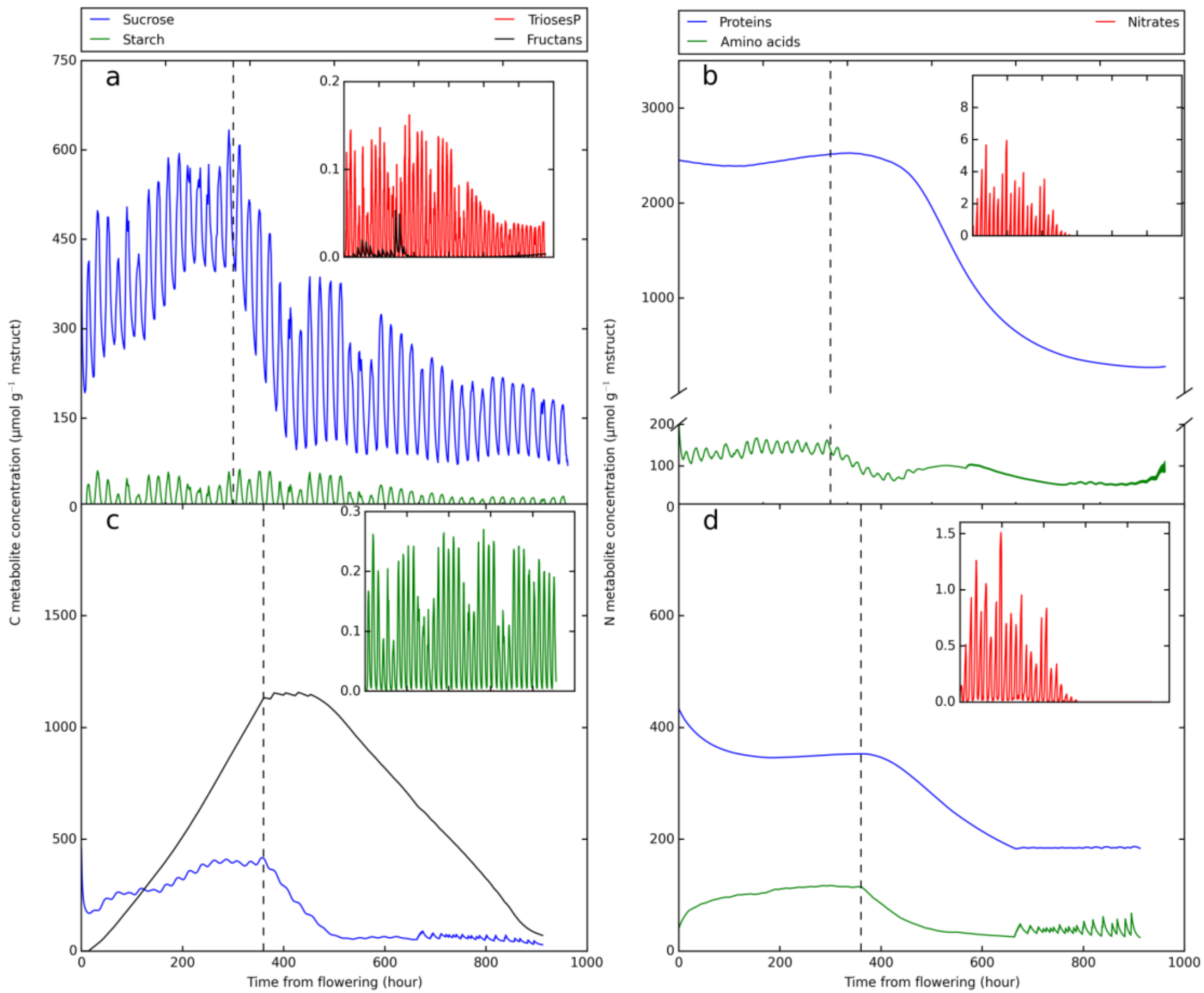
$$\begin{pmatrix} \frac{dc_1}{dt} \\ \frac{dc_2}{dt} \\ \vdots \\ \frac{dc_m}{dt} \end{pmatrix} = \begin{pmatrix} p_1(t, \boldsymbol{c}) \\ p_2(t, \boldsymbol{c}) \\ \vdots \\ p_m(t, \boldsymbol{c}) \end{pmatrix}$$

avec $\boldsymbol{c}$ un vecteur tel que $\boldsymbol{c}(t) = [c_1(t), c_2(t), \ldots, c_m(t)]$, et $p_1, p_2, \cdots, p_m$ les fonctions représentant les processus physiologiques.

3. Application à la modélisation biologique du blé : le modèle à compartiments « CN-Wheat »

- **Utilisation de SciPy.Integrate pour calculer les concentrations à un pas de temps donné**
  - Création de la liste de conditions initiales $\boldsymbol{c}(t_0)=[c_1(t_0), c_2(t_0), \ldots, c_m(t_0)]$ à partir de la valeur courante des concentrations de chaque métabolite.
  - Définition d'une fonction , $\boldsymbol{P}=\frac{d\,\boldsymbol{c}}{dt}$ calculant les dérivées $[\frac{dc_1}{dt}, \frac{dc_2}{dt}, \cdots, \frac{dc_m}{dt}]$ à partir d'un temps t et de conditions initiales $\boldsymbol{c}(t)$, $t \in [t_{0,} t_i]$
  - Appelle de la fonction **scipy.integrate.odeint(...)**, avec :
    - en 1er argument la fonction $\boldsymbol{P}$,
    - en 2d argument les conditions initiales $\boldsymbol{c}(t_0)$,
    - et en 3e argument le temps t_i auquel on veut calculer les concentrations
  - Quand le calcul est terminé, **scipy.integrate.odeint(...)** renvoie les concentrations aux temps t_0 et t_i

3. Application à la modélisation biologique du blé : le modèle à compartiments « CN-Wheat »



Concentration des métabolites (R.Barillot, iCROPM2016)

4. Pour se détendre...

Des renards et des lapins

Modèle proie-prédateur : dynamique de systèmes biologiques dans lesquels un prédateur et sa proie interagissent (https://fr.wikipedia.org/wiki/%C3%89quations_de_Lotka-Volterra)

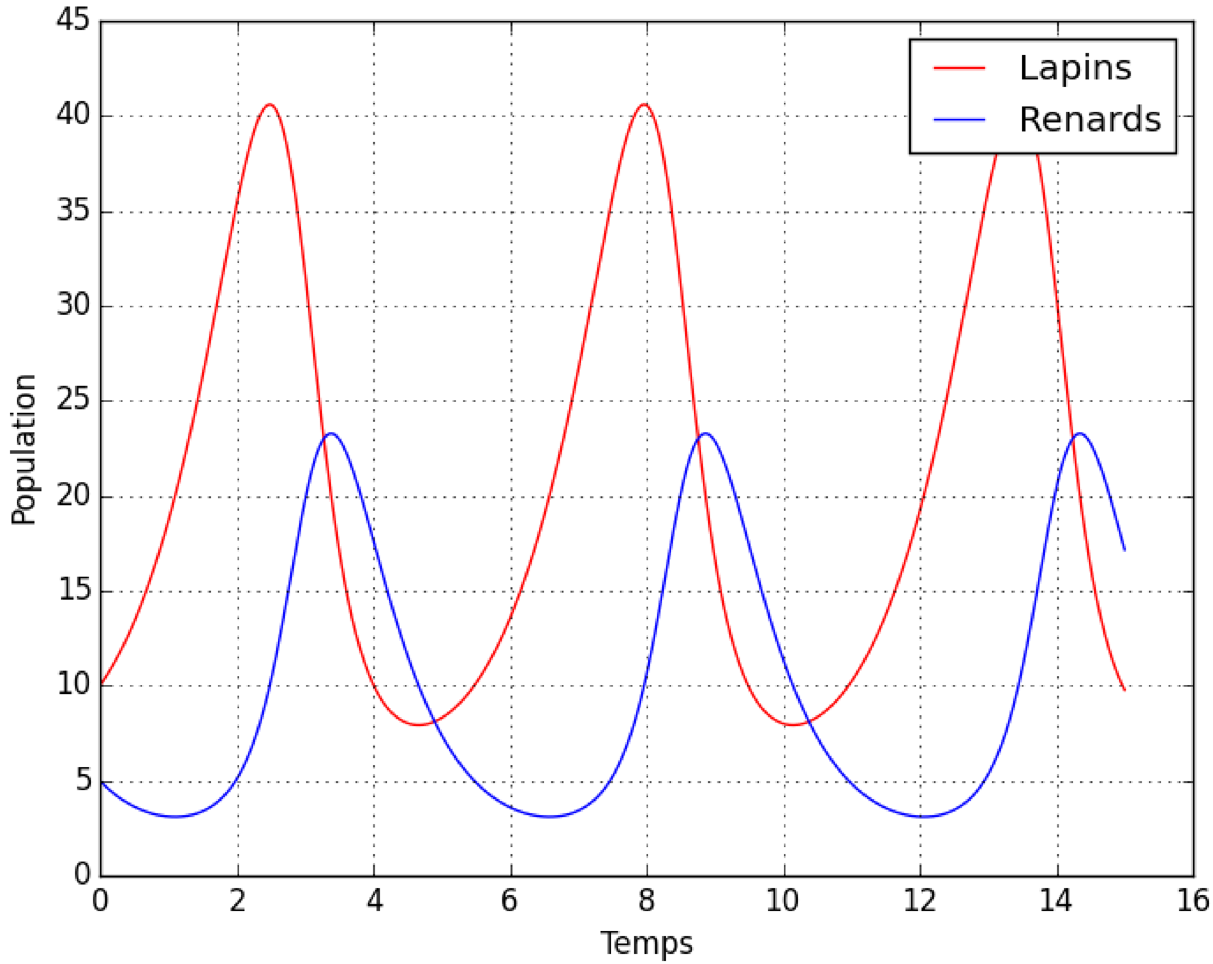
$$\begin{pmatrix} \frac{du}{dt} \\ \frac{dv}{dt} \end{pmatrix} = \begin{pmatrix} A \cdot u - B \cdot u \cdot v \\ -c \cdot u + D \cdot B \cdot u \cdot v \end{pmatrix}$$

avec :

- u et v des variables fonction du temps
  - u nombre de lapins,
  - v nombre de renards,
- et A, B, C et D des paramètres (constants) définissant la dynamique de la population :
  - A taux de reproduction des lapins quand il n'y a pas de renard,
  - B taux de mortalité des lapins dû aux renards,
  - C taux de mortalité des renards quand il n'y a pas de lapin,
  - D taux de reproduction des renards en fonction des lapins rencontrés et mangés.

4. Pour se détendre...

- Des renards et des lapins (suite)



Modèle proie-prédateur : dynamique des populations de renards et de lapins
http://scipy-cookbook.readthedocs.io/items/LoktaVolterraTutorial.html

4. Pour se détendre...

- Une invasion de zombies (d'après http://mysite.science.uottawa.ca/rsmith43/Zombies.pdf)

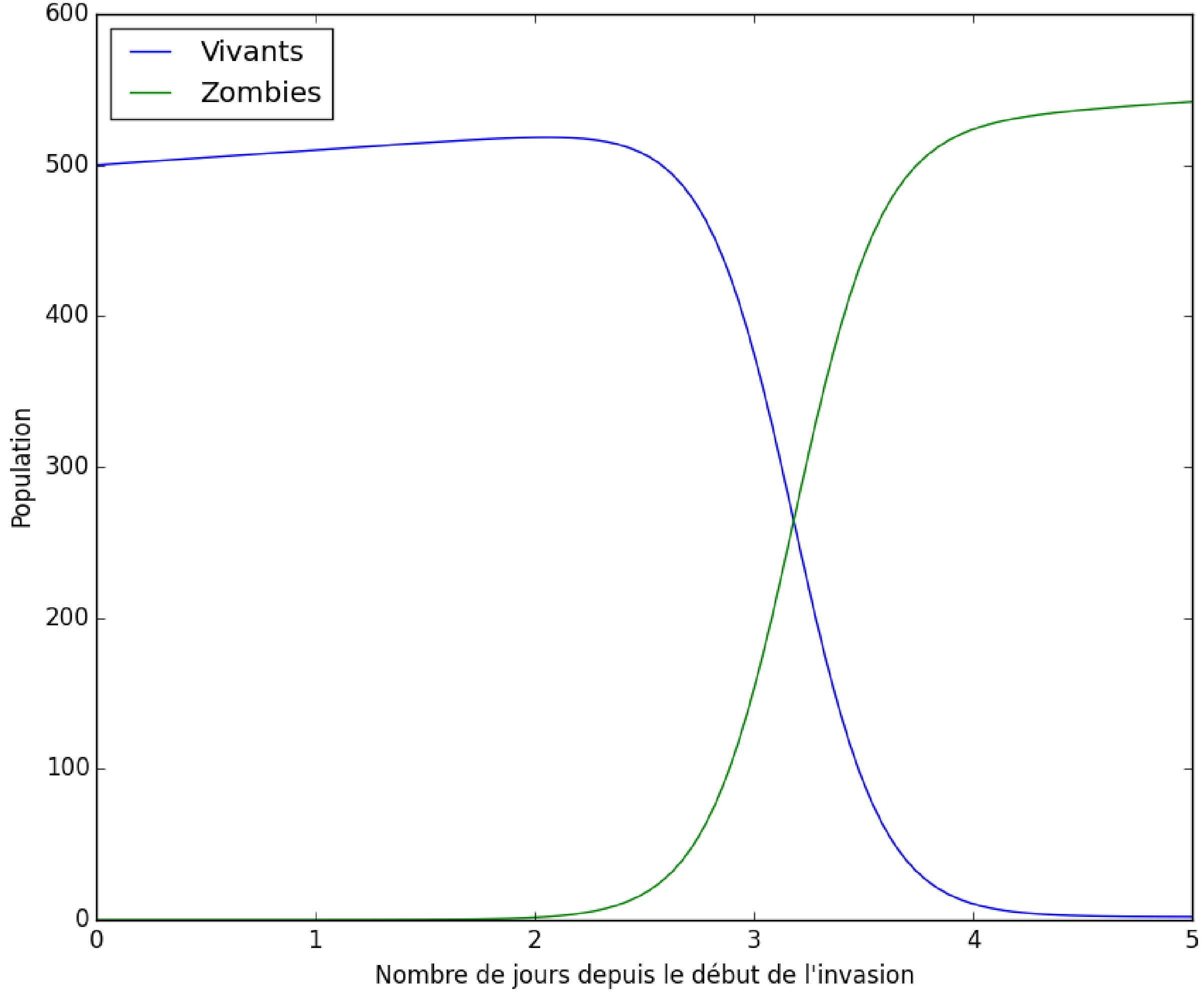
$$\begin{pmatrix} \frac{ds}{dt} \\ \frac{dz}{dt} \\ \frac{dr}{dt} \end{pmatrix} = \begin{pmatrix} P - B \cdot s \cdot z - D \cdot s \\ B \cdot s \cdot z + G \cdot r - A \cdot s \cdot z \\ D \cdot s + A \cdot s \cdot z - G \cdot r \end{pmatrix}$$

avec :

- s, z et r des variables fonction du temps :
  - s : nombre de victimes potentielles,
  - z : nombre de zombies,
  - r : nombre de personnes « tuées » par un zombie,
- et P, D, B, G et A des paramètres (constantes) définissant la dynamique de la population :
  - P : taux de natalité de la population,
  - D : taux de mortalité de la population,
  - B : la probabilité qu'une personne vivante devienne un zombie,
  - G : la probabilité qu'une personne décédée soit ressuscitée en zombie,
  - A : la probabilité qu'un zombie soit « tué »

4. Pour se détendre...

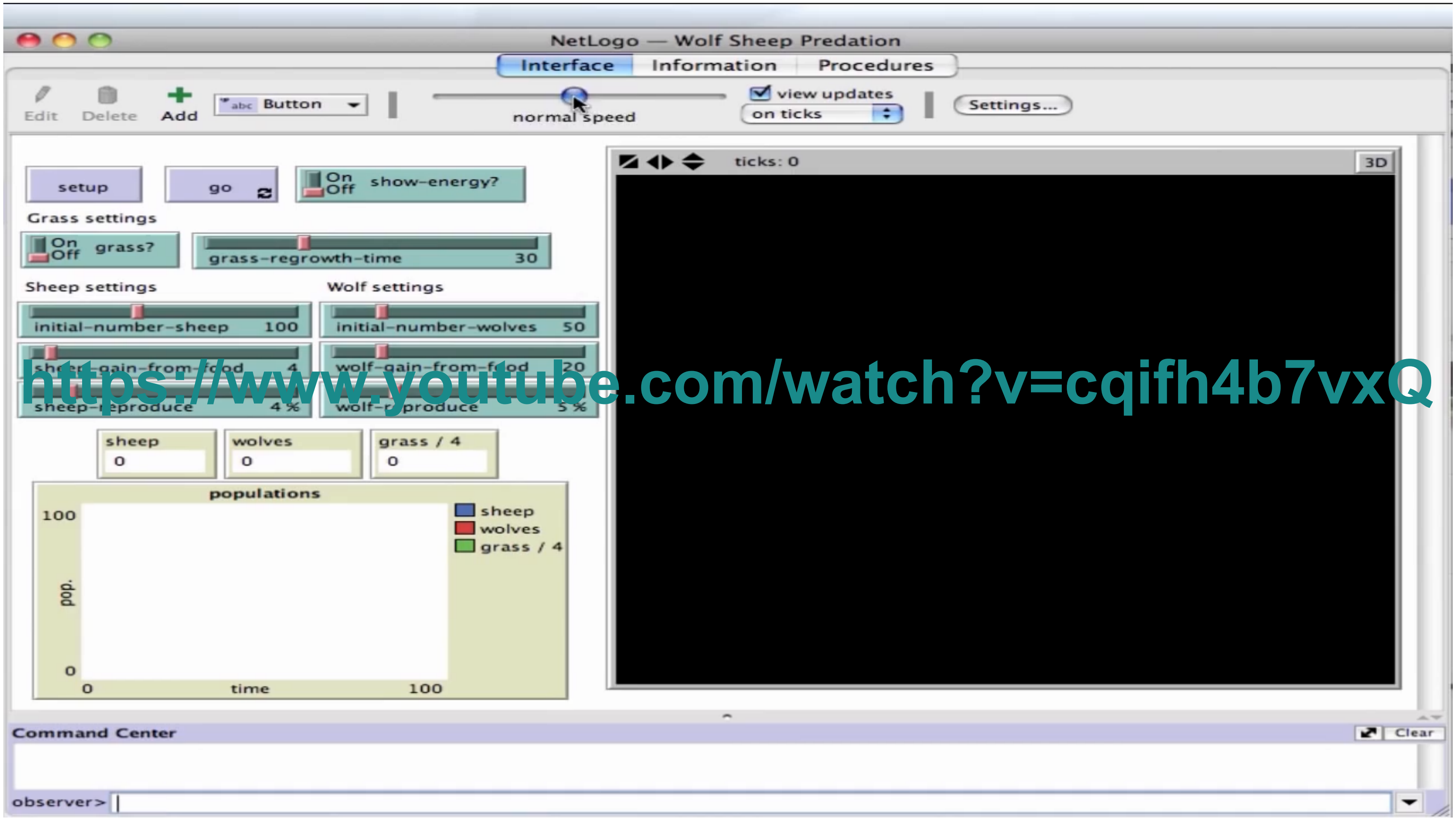
- Une invasion de zombies (suite)



Modèle d'invasion de Zombies : dynamique des populations de vivants et de zombies
http://scipy-cookbook.readthedocs.io/items/Zombie_Apocalypse_ODEINT.html

4. Pour se détendre...

- Des loups et des moutons



https://www.youtube.com/watch?v=cqifh4b7vxQ

Modèle proie-prédateur (moutons et loups) de la bibliothèque NetLogo

Merci de votre attention