M2I BBS

# Modélisation et Simulation de Systèmes Biologiques

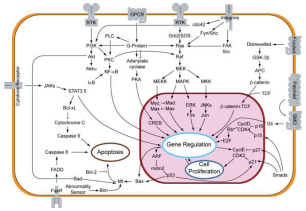
(#1 - intro math)

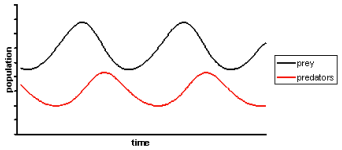
Georges Czaplicki, UPS / IPBS-CNRS
Tél. : 05.61.17.54.04, email : cgeorge@ipbs.fr

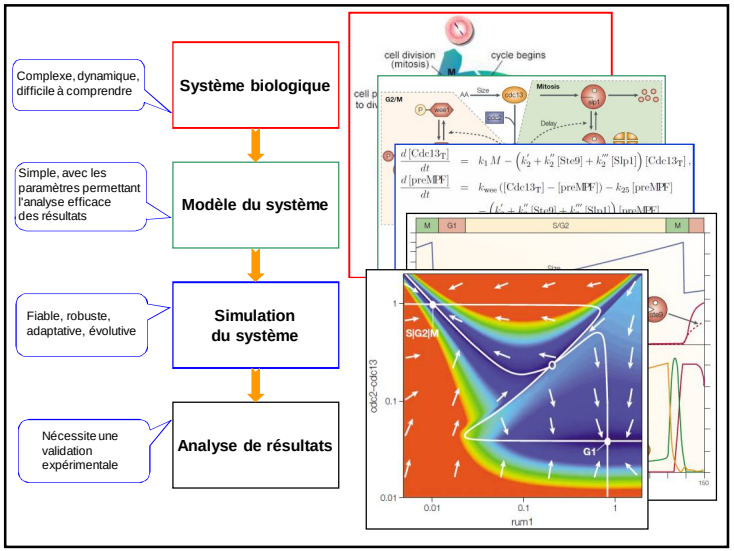
## Modélisation dans la biologie des systèmes

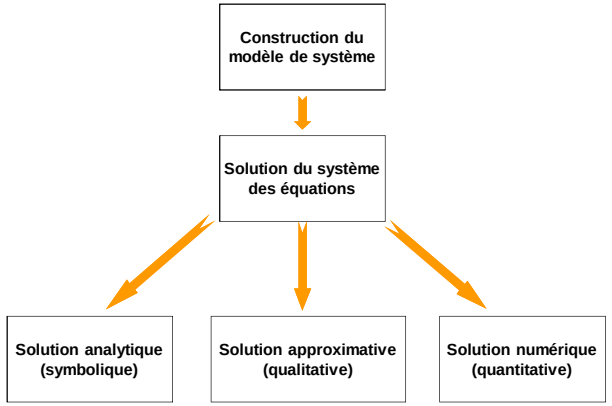
La biologie systémique s'appuie lourdement sur l'utilisation d'outils mathématiques et de ressources bioinformatiques pour la modélisation et la simulation de systèmes biologiques complexes et dynamiques.

Les méthodes d'analyse développées pour les besoins de la simulation couvrent beaucoup de domaines différents et sont valides à des niveaux différents de l'organisation de systèmes biologiques. Les approches pareilles peuvent être utilisées pour analyser les processus dynamiques au niveau de molécules ou de cellules, mais aussi au niveau de populations entières.



*Objectif :*
**Apprendre les bases de la modélisation de processus biologiques.**

*Contenu :*

- **Introduction aux outils mathématiques**
- **Construction de modèles dynamiques dans biologie**
- **Recherche de solutions analytiques**
  - **présentation des équations différentielles**
  - **méthodes d'obtention de solutions analytiques**
- **Théorie qualitative de systèmes dynamiques**
  - **simplification de systèmes d'équations**
  - **points stationnaires et leur stabilité**
  - **portrait de phase et trajectoires**
- **Approches numériques**
  - **introduction au calcul numérique et au langage Matlab**
  - **intégration de jeux d'équations différentielles**
- **Sélection des applications pratiques**
  - **cinétique enzymatique (réactions chimiques)**
  - **modélisation de la croissance de la biomasse**
  - **processus périodiques et systèmes prédateur-proie**
  - **sélection d'un des espèces équivalents**
  - **les *triggers* génétiques (biosynthèse)**

## Introduction aux outils mathématiques

- **Calcul différentiel**
  - **fonctions**
  - **dérivées**
  - **règles de différentiation**
  - **dérivées partielles**
  - **développement en série de Taylor**
- **Calcul intégral**
  - **définitions**
  - **règles d'intégration**
- **Algèbre linéaire**
  - **vecteurs et matrices**
  - **opérations matricielles**
- **Equations différentielles**
  - **types, ordres**
  - **exemples**

## Dérivation

**Functions**

Some relationships that appear in this book are functions of a single variable. These functions relate the value of one variable to the value of another. The standard notation is $y = f(x)$ where $x$ is the input variable (or *argument*), $y$ is the output (the function value), and $f(\cdot)$ is the function itself. We also work with functions of multiple variables. Such functions each take a group of numbers as input, and return a single number as output e.g. $y = g(x_1, x_2, x_3)$, where the input argument is the set of numbers $x_1$, $x_2$, and $x_3$.

**The derivative**

The *derivative* of a function is the rate of change of the function value at a point (i.e. at a particular value of the input variable). To define the derivative of a function of a single variable, we first consider the average rate of change between two points (Figure B.1). The rate of change of a function $f(\cdot)$ between the two points $x_0$ and $x$ is the ratio of the change in the function value to the change in the argument, as follows:

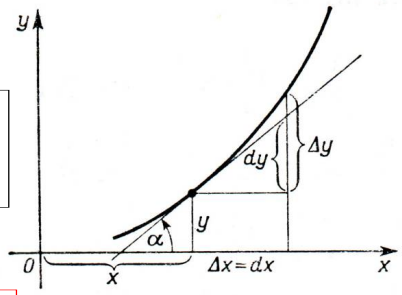
$$\text{Average rate of change:} \quad \frac{f(x) - f(x_0)}{x - x_0}. \tag{B.1}$$

## Définition de la dérivée d'une fonction

Afin de faciliter l'analyse de fonctions, on introduit la notion de la dérivée. La dérivée *y'(x)* d'une fonction *y=f(x)* qui dépend d'une seule variable *x*, est définie comme suit :

$$y' = \frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{y(x+\Delta x) - y(x)}{\Delta x}$$

La dérivée montre la variation de la fonction *y=f(x)* quand on varie son argument x. Voici l'interprétation géométrique de la dérivée :




$$\tan(\alpha) = \frac{\Delta y}{\Delta x} \xrightarrow{\Delta x \to 0} \frac{dy}{dx} = y'(x)$$

## Calcul de dérivées

La dérivée peut être calculée d'après sa définition :

$$y = f(x) = x$$

$$y' = \lim_{\Delta x \to 0} \frac{(x+\Delta x)-(x)}{\Delta x} = \lim_{\Delta x \to 0} \frac{\Delta x}{\Delta x} = 1$$

$$y = f(x) = a \cdot x + b$$

$$y' = \lim_{\Delta x \to 0} \frac{[a(x+\Delta x)+b]-[a(x)+b]}{\Delta x} = \lim_{\Delta x \to 0} \frac{a \cdot \Delta x}{\Delta x} = a$$

$$y = f(x) = x^2$$

$$y' = \lim_{\Delta x \to 0} \frac{(x+\Delta x)^2 - x^2}{\Delta x} = \lim_{\Delta x \to 0} \frac{(x^2 + 2x\Delta x + \Delta x^2) - x^2}{\Delta x} = \lim_{\Delta x \to 0} \frac{2x\Delta x + \Delta x^2}{\Delta x} = \lim_{\Delta x \to 0} (2x + \Delta x) = 2x$$

On peut aussi démontrer la validité de la relation :

$$y = x^n \quad \Rightarrow \quad y' = n \cdot x^{n-1}$$

d'où on obtient, par exemple, la dérivée de la racine carrée :

$$y = \sqrt{x} = x^{\frac{1}{2}} \quad \Rightarrow \quad y' = \frac{1}{2} x^{-\frac{1}{2}} = \frac{1}{2\sqrt{x}}$$

## Dérivées d'ordre supérieur

La dérivée du deuxième ordre ***y''=f ''(x)*** est calculée comme la dérivée de la première dérivée ***y'=f '(x)*** d'une fonction ***y=f(x)*** :

$$y'' = \frac{d}{dx}\left(\frac{dy}{dx}\right) = \frac{dy'}{dx} = \frac{d^2 y}{dx^2}$$

On applique donc la définition deux fois. Exemples :

$$y = f(x) = ax + b$$
$$y' = a$$
$$y'' = 0$$

$$y = f(x) = ax^2 + bx + c$$
$$y' = 2ax + b$$
$$y'' = 2a$$

En général, la dérivée d'ordre ***n*** est la dérivée de la dérivée d'ordre ***n-1***

$$y^{(n)} = \frac{dy^{(n-1)}}{dx} = \frac{d^n y}{dx^n}$$

est on l'obtient en appliquant la définition ***n*** fois.

## Règles basiques de différentiation

1. **Constant function**

$$\frac{d}{dx}c = 0$$

2. **Identity function**

$$\frac{d}{dx}x = 1$$

3. **Power function**

$$\frac{d}{dx}x^n = nx^{n-1}$$

4. **Exponential function**

$$\frac{d}{dx}e^x = e^x$$

5. **Natural logarithmic function**

$$\frac{d}{dx}\ln(x) = \frac{1}{x}$$

6. **Scalar multiplication**

$$\frac{d}{dx}c\,f(x) = c\,\frac{d}{dx}f(x)$$

7. **Addition**

$$\frac{d}{dx}\left(f_1(x) + f_2(x)\right) = \frac{d}{dx}f_1(x) + \frac{d}{dx}f_2(x)$$

8. **Product rule**

$$\frac{d}{dx}\left(f_1(x)f_2(x)\right) = f_1(x)\left(\frac{d}{dx}f_2(x)\right) + f_2(x)\left(\frac{d}{dx}f_1(x)\right)$$

9. **Quotient rule**

$$\frac{d}{dx}\left(\frac{f_1(x)}{f_2(x)}\right) = \frac{f_2(x)\left(\frac{d}{dx}f_1(x)\right) - f_1(x)\left(\frac{d}{dx}f_2(x)\right)}{[f_2(x)]^2}$$

10. **Chain rule**

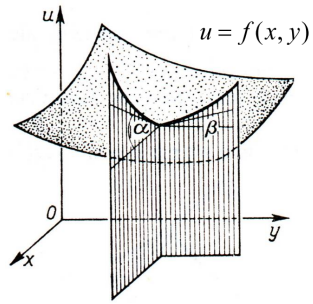
$$\frac{d}{dx}f_1(f_2(x)) = \left(\frac{d}{dw}f_1(w)\Big|_{w=f_2(x)}\right)\frac{d}{dx}f_2(x)$$

**Les dérivées partielles :**

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\Delta x \to 0}\frac{f(x+\Delta x, y) - f(x,y)}{\Delta x}$$

$$\frac{\partial f(x,y)}{\partial y} = \lim_{\Delta y \to 0}\frac{f(x, y+\Delta y) - f(x,y)}{\Delta y}$$

**L'interprétation géométrique des dérivées partielles**




$$\frac{\partial u}{\partial x} = \tan\alpha$$

$$\frac{\partial u}{\partial y} = \tan\beta$$

## Développement en série de Taylor

$$f(x) \approx f(x_0) + A_1\cdot(x-x_0) + A_2\cdot(x-x_0)^2 + A_3\cdot(x-x_0)^3 + \ldots + A_k\cdot(x-x_0)^k + \ldots$$

$$A_1 = \frac{df(x_0)}{dx}$$

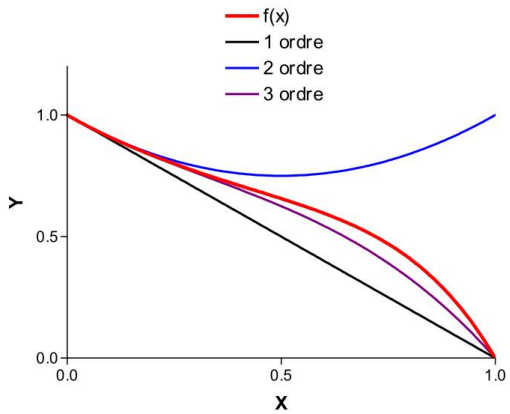
$$A_2 = \frac{1}{2}\frac{d^2 f(x_0)}{dx^2}$$

$$A_3 = \frac{1}{6}\frac{d^3 f(x_0)}{dx^3}$$

$$\ldots$$

$$A_k = \frac{1}{k!}\frac{d^k f(x_0)}{dx^k}$$

Les approximations successives d'une fonction :




## Les fonctions de plusieurs variables

$$
\begin{aligned}
f(x,y) \approx\ & f(x_0,y_0) + \frac{\partial f(x_0,y_0)}{\partial x_0}\cdot(x-x_0) \\
& + \frac{\partial f(x_0,y_0)}{\partial y_0}\cdot(y-y_0) \\
& + \frac{1}{2}\frac{\partial^2 f(x_0,y_0)}{\partial x_0^2}\cdot(x-x_0)^2 \\
& + \frac{1}{2}\frac{\partial^2 f(x_0,y_0)}{\partial y_0^2}\cdot(y-y_0)^2 \\
& + \frac{\partial^2 f(x_0,y_0)}{\partial x_0 \partial y_0}\cdot(x-x_0)\cdot(y-y_0) \\
& + \ldots
\end{aligned}
$$

## Définition de l'intégrale d'une fonction

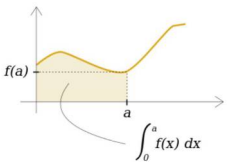
### Calcul d'aire

Dans un plan muni d'un repère cartésien, on choisit comme unité d'aire, l'aire du quadrilatère OIKJ où O est l'origine du repère et I, J et K les points de coordonnées respectives (1 ; 0), (0 ; 1) et (1 ; 1).

Si $f$ est une fonction réelle positive continue prenant ses valeurs dans un segment $I = [a, b]$, alors l'**intégrale** de $f$ sur $I$, notée

$$\int_{x\in I} f(x)\,\mathrm{d}x,$$

est l'aire d'une surface délimitée par la représentation graphique de $f$ et par les trois droites d'équation $x = a$, $x = b$, $y = 0$, surface notée $S_f$. (Voir schéma ci-contre pour l'intervalle $I = [0, a]$.)



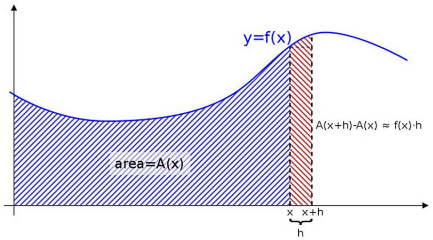

Représentation graphique d'un intégrande $f$ positif et de son intégrale.

On appelle $f$ un **intégrande**[2], et on note ∫ (un s allongé, mis pour somme) l'opérateur mathématique, appelé intégrateur, qui est associé à l'intégration. Ce symbole est un ancien *s* long : en effet, Leibniz s'est servi de l'initiale du mot latin *summa*, « somme », lequel était le plus souvent écrit *ſumma*. À la différence du *s* long, ſ, en typographie, garde toujours une hampe descendant au-dessous de la ligne de base, en romaine comme en italique. Voir l'article Notation de Leibniz pour une justification de la notation complète, et en particulier du symbole d$x$.

Dérivée et intégrale = fonctions inverses

$$A(x) = \int f(x)\,dx$$

A = l'aire sous la courbe f(x)
(ici, entre 0 et la coordonnée x)




L'aire entre x et x+h = A(x+h) – A(x)

L'aire entre x et x+h = h · f(x)

$$h \cdot f(x) \approx A(x+h) - A(x)$$

$$f(x) \approx \frac{A(x+h)-A(x)}{h} \longrightarrow f(x) = \frac{dA}{dx}$$

## Propriétés des intégrales

Soient $f$ et $g$ deux fonctions continues sur $I$ et $a$, $b$ et $c$ trois réels de $I$.

- $\int_a^a f(x)\,\mathrm{d}x = 0$
- $\int_b^a f(x)\,\mathrm{d}x = -\int_a^b f(x)\,\mathrm{d}x$
- $\int_a^b f(x)\,\mathrm{d}x + \int_b^c f(x)\,\mathrm{d}x = \int_a^c f(x)\,\mathrm{d}x$ (relation de Chasles)
- $\forall \lambda \in \mathbb{R}, \int_a^b \lambda f(x)\,\mathrm{d}x = \lambda \int_a^b f(x)\,\mathrm{d}x$
- $\int_a^b (f(x)+g(x))\,\mathrm{d}x = \int_a^b f(x)\,\mathrm{d}x + \int_a^b g(x)\,\mathrm{d}x$ (linéarité de la fonction intégrale)
- Si $f(x) \leq g(x)$ sur $[a, b]$, alors $\int_a^b f(x)\,\mathrm{d}x \leq \int_a^b g(x)\,\mathrm{d}x$.
- **Inégalité de la moyenne**. Si $f$ est continue sur $[a, b]$, avec $a \leq b$ et si pour tout $x$ de cet intervalle, on a :

$$m \leq f(x) \leq M,$$

alors

$$m\,(b-a) \leq \int_a^b f(x)\,\mathrm{d}x \leq M\,(b-a).$$

**Exemple : moyenne d'une fonction**

**Définition : Valeur moyenne d'une fonction** :

Soit $f$ une fonction *définie* et *continue* sur un intervalle $[a\,;b]$ tel que $a < b$.
Alors, on appelle ***valeur moyenne de la fonction f*** sur l'intervalle $[a\,;b]$, qu'on note $\mu$ ou $\mu(f)$ ou parfois $\mu_{[a;b]}(f)$ le ***nombre réel*** défini comme suit :

$$\mu = \frac{1}{b-a}\int_a^b f(x)\,dx \quad (\textit{lire} \text{ « } \textbf{mu} \text{ »}).$$

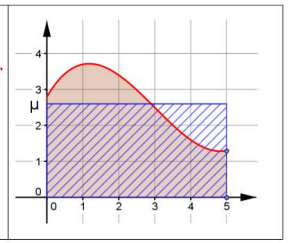


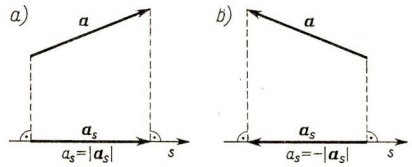

TABLEAU DES PRIMITIVES

Soient $u$ et $v$ deux fonctions ***définies*** sur un intervalle $I$ où elles sont ***continues***.

| ***Fonctions usuelles*** | | ***Fonctions composées*** | |
|---|---|---|---|
| Fonctions | Primitives | Fonctions | Primitives |
| $0$ | $k$ | $u'+v'$ | $u+v+C$ |
| $a$ | $ax+C$ | $k\,u'$ | $k\,u+C$ |
| $x$ | $\frac{x^2}{2}+C$ | $u'u$ | $\frac{u^2}{2}+C$ |
| $x^n, n\neq -1$ $n\in\mathbb{Z}$ | $\frac{x^{n+1}}{n+1}+C$ | $u'u^n, n\neq -1$ $n\in\mathbb{Z}$ | $\frac{u^{n+1}}{n+1}+C$ |
| $\frac{1}{\sqrt{x}}$ | $2\sqrt{x}+C$ | $\frac{u'}{\sqrt{u}}$ $u>0.$ | $2\sqrt{u}+C$ , |
| $\frac{1}{x^2}$ | $-\frac{1}{x}+C$ | $\frac{u'}{u^2}$ | $\frac{-1}{u}+C$ |
| $e^x$ | $e^x+C$ | $u'e^u$ | $e^u+C$ |
| $\frac{1}{x}$ , $x>0$ | $\ln x+C$ | $\frac{u'}{u}$ , $u>0$ | $\ln u+C$ |
| $\sin x$ | $-\cos x+C$ | $u'\sin u$ | $-\cos u+C$ |
| $\cos x$ | $\sin x+C$ | $u'\cos u$ | $\sin u+C$ |

**Rappel des éléments d'algèbre linéaire**

1. Vecteurs :




a) coordonnées

$$\mathbf{a} = (a_x, a_y, a_z)$$

b) norme (longueur)

$$|a| = \sqrt{a_x^2 + a_y^2 + a_z^2} = \sqrt{\sum_{i=1}^{3} a_i^2}$$

c) opérations sur vecteurs

- égalité

$$\mathbf{a} = (a_x, a_y, a_z), \qquad \mathbf{b} = (b_x, b_y, b_z)$$

$$\mathbf{a} = \mathbf{b} \quad \text{uniquement si} \quad a_x = b_x, \quad a_y = b_y, \quad a_z = b_z$$

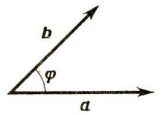
- produit d'un scalaire et un vecteur

$$r\mathbf{a} = (ra_x, ra_y, ra_z)$$

- somme de deux vecteurs

$$\mathbf{a} \pm \mathbf{b} = (a_x \pm b_x, a_y \pm b_y, a_z \pm b_z)$$

- produit scalaire de deux vecteurs




$$\mathbf{a} \cdot \mathbf{b} = |a| \cdot |b| \cdot \cos(\varphi)$$

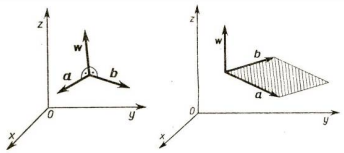
$$\mathbf{a} \cdot \mathbf{b} = a_x b_x + a_y b_y + a_z b_z = \sum_{i=1}^{3} a_i b_i$$

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{b} \cdot \mathbf{a}$$

$$\mathbf{a} \cdot (\mathbf{b} + \mathbf{c}) = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \cdot \mathbf{c}$$

$$\mathbf{a} \cdot \mathbf{b} = 0 \Leftrightarrow \varphi = 90°$$

- produit vectoriel de deux vecteurs




$$\mathbf{w} = \mathbf{a} \times \mathbf{b}$$

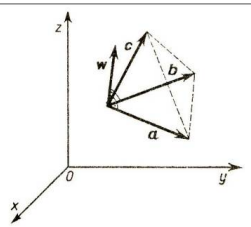
$$|w| = |a| \cdot |b| \cdot \sin(\varphi)$$

$$\mathbf{a} \times \mathbf{b} = -\mathbf{b} \times \mathbf{a}$$

$$\mathbf{a} \times (\mathbf{b} + \mathbf{c}) = \mathbf{a} \times \mathbf{b} + \mathbf{a} \times \mathbf{c}$$

$$S = |w| = |\mathbf{a} \times \mathbf{b}|$$

- produit mixte




$$(\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c}$$

$$(\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c} = (\mathbf{b} \times \mathbf{c}) \cdot \mathbf{a} = (\mathbf{c} \times \mathbf{a}) \cdot \mathbf{b}$$

$$(\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c} = -(\mathbf{c} \times \mathbf{b}) \cdot \mathbf{a}$$

$$V = \frac{1}{6} |(\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c}| = \frac{1}{6} \det \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{bmatrix}$$

- angle entre deux vecteurs

$$\cos(\varphi) = \frac{\mathbf{a} \cdot \mathbf{b}}{|a||b|} = \frac{a_x b_x + a_y b_y + a_z b_z}{\sqrt{a_x^2 + a_y^2 + a_z^2}\sqrt{b_x^2 + b_y^2 + b_z^2}} = \frac{\sum_{i=1}^{3} a_i b_i}{\sqrt{\sum_{i=1}^{3} a_i^2}\sqrt{\sum_{i=1}^{3} b_i^2}}$$

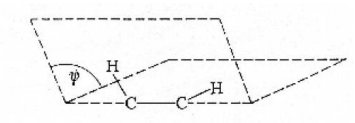
- dépendance linéaire des vecteurs

$$\lambda_1 a_1 + \lambda_2 a_2 + \ldots + \lambda_n a_n = 0, \qquad \lambda_i \neq 0$$

- vecteurs de base (n pour l'espace n-dimensionnelle, linéairement indépendants)

$$\mathbf{e_1} = (1,0,0) \quad \mathbf{e_2} = (0,1,0) \quad \mathbf{e_3} = (0,0,1)$$
$$\mathbf{v} = v_1\mathbf{e_1} + v_2\mathbf{e_2} + v_3\mathbf{e_3}$$

d) exemple : angle dièdre




2. Matrices

a) définition d'une matrice rectangulaire

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \ldots a_{1m} \\ a_{21} & a_{22} \ldots a_{2m} \\ \ldots & \ldots \\ a_{n1} & a_{n2} \ldots a_{nm} \end{bmatrix}$$

b) cas spéciaux

| | |
|---|---|
| matrice carrée | $\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & j \end{bmatrix} \qquad n = m$ |
| matrice diagonale | $\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix}$ |
| matrice unitaire (**1**) | $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ |

| | |
|---|---|
| matrice zéro (**0**) | $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ |
| matrice symétrique | $\begin{bmatrix} a & d & e \\ d & b & f \\ e & f & c \end{bmatrix} \qquad A_{ij} = A_{ji}$ |
| matrice transposée<br>$(A^T)^T = A$<br>$(AB)^T = B^T A^T$<br>$(A+B)^T = A^T + B^T$ | $\mathbf{A} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & j \end{bmatrix} \qquad \mathbf{A^T} = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & j \end{bmatrix}$ |

c) opérations matricielles

- égalité

$$\mathbf{A} = \mathbf{B} \Leftrightarrow A_{ij} = B_{ij} \qquad (i = 1,\ldots,n \qquad j = 1,\ldots,m)$$

- somme de deux matrices

| | |
|---|---|
| $\mathbf{C} = \mathbf{A} + \mathbf{B} \qquad C_{ij} = A_{ij} + B_{ij}$ | $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 4 & -2 & 1 \\ 3 & 0 & -1 \end{bmatrix}$ <br> $\mathbf{A}+\mathbf{B} = \begin{bmatrix} 1+4 & 2-2 & 3+1 \\ 0+3 & 1+0 & 4-1 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 4 \\ 3 & 1 & 3 \end{bmatrix}$ <br> $\mathbf{A}-\mathbf{B} = \begin{bmatrix} 1-4 & 2-(-2) & 3-1 \\ 0-3 & 1-0 & 4-(-1) \end{bmatrix} = \begin{bmatrix} -3 & 4 & 2 \\ -3 & 1 & 5 \end{bmatrix}$ |

- produit de deux matrices

$$\mathbf{C} = \mathbf{A} \cdot \mathbf{B}, \qquad C_{ij} = A_{i1}B_{1j} + A_{i2}B_{2j} + \ldots + A_{in}B_{nj} = \sum_{k=1}^{n} A_{ik}B_{kj} = \mathbf{a_i} \cdot \mathbf{b_j}$$

$$(\mathbf{AB} \neq \mathbf{BA})$$

**Exemple 1 :**

$$A = \begin{bmatrix} 4 & 2 & 1 \\ 3 & 5 & 2 \\ 0 & 1 & 2 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 2 & 4 & 6 \end{bmatrix}$$

$$A \cdot B = \begin{bmatrix} 12 & 16 & 20 \\ 22 & 24 & 26 \\ 7 & 10 & 13 \end{bmatrix} \qquad B \cdot A = \begin{bmatrix} 10 & 15 & 11 \\ 18 & 17 & 9 \\ 20 & 30 & 22 \end{bmatrix}$$

**Exemple 2 :**

$A_n^m \rightarrow$ matrice avec $m$ lignes et $n$ colonnes

$$A_n^m \cdot B_k^n = C_k^m \qquad \text{Ex.}: A_3^4 \cdot B_2^3 = C_2^4$$

$$A = \begin{bmatrix} 1 & 0 & 2 \\ 3 & 4 & 1 \\ 4 & 5 & 2 \\ 0 & 2 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 1 \\ 3 & 4 \\ 5 & 2 \end{bmatrix}$$

$$A \cdot B = \begin{bmatrix} 12 & 5 \\ 23 & 21 \\ 33 & 28 \\ 26 & 16 \end{bmatrix} \qquad B \cdot A = \begin{bmatrix} \text{??? Error : Inner matrix} \\ \text{dimensions must agree} \end{bmatrix}$$

- rang d'une matrice

***Le nombre maximal des vecteurs (lignes, colonnes) linéairement indépendants. (Opérations permises : changement d'ordre, d'échelle et combinaison linéaire)***

d) déterminant d'une matrice

- définition

**det(A) = |A|** ***: une fonction de tous les éléments de la matrice A***

- propriétés

$$\det(\mathbf{A}) = \det(\mathbf{A}^{\mathbf{T}})$$

det(**A**) = 0 s'il y a des lignes (colonnes) linéairement dépendantes
det(**A**) se calcule aisément d'après le développement de Laplace

e) matrice inverse

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I} \qquad \mathbf{B} = \mathbf{A}^{-1}$$

$$A = \begin{bmatrix} 1 & 0 & 2 \\ 3 & 4 & 1 \\ 4 & 5 & 2 \end{bmatrix} \qquad B = \begin{bmatrix} 3 & 10 & -8 \\ -2 & -6 & 5 \\ -1 & -5 & 4 \end{bmatrix} \qquad A \cdot B = B \cdot A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

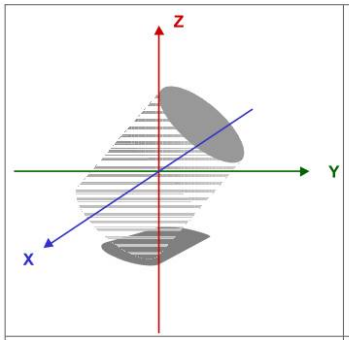
f) le polynôme caractéristique

$$\det(\mathbf{A} - \lambda\mathbf{I}) = |\mathbf{A} - \lambda\mathbf{I}| = 0$$

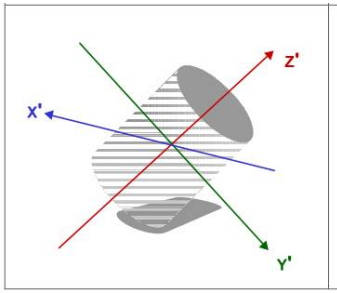
$$|\mathbf{A} - \lambda\mathbf{I}| = \begin{vmatrix} a_{11} - \lambda & a_{12} & a_{13} \\ a_{21} & a_{22} - \lambda & a_{23} \\ a_{31} & a_{32} & a_{33} - \lambda \end{vmatrix} = 0$$

Les racines de cette équation représentent les valeurs propres du système.

g) diagonalisation, valeurs et vecteurs propres




$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$




$$\begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}$$

$$\mathbf{A} = \mathbf{VDV}^{-1} \qquad \mathbf{D} = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}, \qquad \mathbf{V} = \begin{pmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{pmatrix}$$

3. Exemple : solution des équations algébriques linéaires

$$\left.\begin{aligned} a_{11}x + a_{12}y + a_{13}z &= b_1 \\ a_{21}x + a_{22}y + a_{23}z &= b_2 \\ a_{31}x + a_{32}y + a_{33}z &= b_3 \end{aligned}\right\} \Rightarrow \quad \mathbf{Ax} = \mathbf{B} \qquad \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \mathbf{B} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

***Solution formelle :*** $\mathbf{Ax} = \mathbf{B} \;\Rightarrow\; \mathbf{A}^{-1}\mathbf{Ax} = \mathbf{A}^{-1}\mathbf{B} \;\Rightarrow\; \mathbf{x} = \mathbf{A}^{-1}\mathbf{B}$

***Règles de Cramer :***

$$x = \frac{\begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{32} & a_{33} \end{vmatrix}}{|\mathbf{A}|} \qquad y = \frac{\begin{vmatrix} a_{11} & b_1 & a_{13} \\ a_{21} & b_2 & a_{23} \\ a_{31} & b_3 & a_{33} \end{vmatrix}}{|\mathbf{A}|} \qquad z = \frac{\begin{vmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ a_{31} & a_{32} & b_3 \end{vmatrix}}{|\mathbf{A}|}$$

***Exemple :***

$$\left.\begin{aligned} 3x + 2y &= 7 \\ 5x - y &= 3 \end{aligned}\right\} \Rightarrow \quad \mathbf{A} = \begin{pmatrix} 3 & 2 \\ 5 & -1 \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} \quad \mathbf{B} = \begin{pmatrix} 7 \\ 3 \end{pmatrix}$$

$$\det(\mathbf{A}) = 3 \cdot (-1) - 2 \cdot 5 = -13 \qquad \mathbf{A}^{-1} = \frac{1}{13}\begin{pmatrix} 1 & 2 \\ 5 & -3 \end{pmatrix}$$

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{B} = \frac{1}{13}\begin{pmatrix} 1 & 2 \\ 5 & -3 \end{pmatrix} \cdot \begin{pmatrix} 7 \\ 3 \end{pmatrix} = \frac{1}{13}\begin{pmatrix} 13 \\ 26 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$x = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$x = \frac{\begin{vmatrix} 7 & 2 \\ 3 & -1 \end{vmatrix}}{-13} = \frac{-13}{-13} = 1 \qquad y = \frac{\begin{vmatrix} 3 & 7 \\ 5 & 3 \end{vmatrix}}{-13} = \frac{-26}{-13} = 2$$

Comment calcule-t-on une fonction d'une matrice ?

$$A \xrightarrow{diag} D = S \cdot A \cdot S^{-1}$$
$$DS = SA$$

$$\boxed{f(A) = S \cdot f(D) \cdot S^{-1}}$$

Exemple :

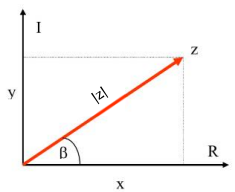
$$D = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} \Rightarrow \quad e^{Dx} = \begin{bmatrix} e^{\lambda_1 x} & 0 & \dots & 0 \\ 0 & e^{\lambda_2 x} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & e^{\lambda_n x} \end{bmatrix}$$

**Math Reminder: complex numbers**

A complex number:

$$\boxed{z = x + i \cdot y}$$

can be graphically represented as follows:




A complex number is fully determined when both of its components ***x*** & ***y*** are known, or when its modulus **|z|** and angle $\beta$ are known:

$$x = |z| \cdot \cos\beta$$
$$y = |z| \cdot \sin\beta$$

Hence, we can write:

$$z = |z| \cdot \cos\beta + i \cdot |z| \cdot \sin\beta = |z| \cdot (\cos\beta + i \cdot \sin\beta)$$

From Euler's formula:

$$e^{i\beta} = \cos\beta + i \cdot \sin\beta$$

we finally get:

$$z = |z| \cdot e^{i\beta}$$

## Equations différentielles : définitions

**Définition 1** ***Equation différentielle ordinaire.*** *Une équation différentielle ordinaire (EDO) est une relation entre la variable réelle $t$, une fonction inconnue $t \mapsto y(t)$ et ses dérivées $y'$, $y''$, ..., $y^{(n)}$ au point $t$ définie par*

$$F(t, y(t), y'(t), y''(t), ..., y^{(n)}(t)) = 0 \text{ (on notera par abus } F(t, y, y', y'', ..., y^{(n)}) = 0) \quad (1.1)$$

*On dit que cette équation est scalaire si $F$ est à valeurs dans $\mathbb{R}$.*
*(N.B. : on pourra utiliser $x$ de temps en temps au lieu de $t$, i.e. $y(t)$ ou $y(x)$)*

**Définition 2** ***Equation différentielle normale.*** *On appelle équation différentielle normale d'ordre $n$ toute équation de la forme*

$$y^{(n)} = f(t, y, y', ..., y^{(n-1)}) \quad (1.2)$$

Donnons un exemple pour mettre les idées au clair.

**Exemple 1** *Equation du premier ordre sous la forme normale*

$$y' = f(t, y) \text{ (ou } \frac{dy}{dt} = f(t, y)) \quad (1.3)$$

Donnons maintenant une classification par linéarité. Une EDO du type (1.1) d'ordre $n$ est linéaire si elle a la forme suivante :

$$a_n(x)y^{(n)} + a_{n-1}(x)y^{(n-1)} + ... + a_1(x)y' + a_0(x)y = g(x), \quad (1.4)$$

noter que
(1) tous les $y^{(i)}$ sont de degré 1, et
(2) tous les coefficients dépendent au plus de $x$

**Exemple 2** *Dire si les équations différentielles suivantes sont linéaires ou non linéaires, et donner leur ordre (on justifiera les réponses).*

$$i.\ (y-x)dx + 4xdy = 0 \quad ii.\ y'' - 2y' + y = 0 \quad iii.\ \frac{d^3y}{dx^3} + x\frac{dy}{dx} - 5y = e^x$$

$$iv.\ (1-y)y' + 2y = e^x \quad v.\ \frac{d^2y}{dx^2} + \sin y = 0 \quad vi.\ \frac{d^4y}{dx^4} + y^2 = 0$$

### 1.3.3 Equations linéaires du premier ordre

**Définition 9** *Une équation différentielle du premier ordre est dite linéaire si elle est linéaire par rapport à la fonction inconnue $y$ et par rapport à sa dérivée $y'$. Une telle équation peut toujours s'écrire sous la forme*

$$A(t)y' + B(t)y = D(t)$$

*On supposera dans toute la suite que $A$, $B$ et $D$ sont continues sur un intervalle $I_0$.*

**Intégration d'une équation linéaire sans second membre**

Il s'agit d'une équation de la forme

$$A(t)y' + B(t)y = 0 \quad (1.10)$$

**Proposition 2** *L'ensemble des solutions de (1.10) sur le domaine $I$ est définie pour tout $t \in I$ par $y(t) = ce^{F(t)}$ avec $F(t) = \int -\frac{B(t)}{A(t)}dt$ où $c$ est une constante.*

**Preuve**. Elle est rapide et simple.

$$Ay' = -By \quad \rightarrow \quad y' = -\frac{B}{A}y \quad \rightarrow \quad \frac{dy}{y} = -\frac{B}{A}dt \quad \rightarrow \quad \int \frac{dy}{y} = -\int \frac{B}{A}dt$$

$$\ln(y) = -\int \frac{B}{A}dt = F + const \quad \rightarrow \quad y = e^{const}.e^F \quad \rightarrow \quad y = Ce^F$$

**Intégration d'une équation linéaire avec second membre**

Considérons l'équation

$$A(t)y' + B(t)y = D(t) \quad (1.12)$$

Désignons par $I \subset I_0$ un intervalle sur lequel $A(t)$ ne possède pas de racine. Soit $y_0$ une intégrale particulière non dégénérée de l'équation sans second membre associée à l'équation (1.12) sur $I$.

**Proposition 3** *L'intégrale générale de l'équation (1.12) sur $I$ est donnée par*

$$y(t) = y_0(t)(\int \frac{D(t)}{y_0(t)A(t)}dt + C)$$

*où $C$ est une constante.*

$$y = C(t)e^{F(t)} \quad \rightarrow \quad y' = C'e^F + Ce^F F' \quad \rightarrow \quad C'e^F - \frac{B}{A}Ce^F$$

$$Ay' + By = D \quad \rightarrow \quad AC'e^F - BCe^F + BCe^F = D$$

$$AC'e^F = D \quad \rightarrow \quad C' = \frac{dC}{dt} = \frac{D}{A}e^{-F}$$

$$C = \int \frac{D}{A}e^{-F}dt + const \quad \rightarrow \quad y = e^F \int \frac{D}{A}e^{-F}dt + const.e^F$$

## Introduction aux équations différentielles

A *differential equation* refers to any equation involving derivatives.

$$\frac{d^2y}{dt^2} - 4\frac{dy}{dt} + 4y = e^{-t} \qquad \frac{dy}{dt} = y - \frac{y^2}{2+\sin t}$$

$$\frac{dy}{dt} = f(t,y) \quad \text{and} \quad y(0) = y_0$$

Separable variables:

The idea is to modify the differential equation algebraically in such a way that all instances of the independent variable are on one side of the equation and all those of the dependent variable are on the other. Then the solution results as the integral of the two sides. For example, consider

$$\frac{dy}{dt} = ay - by^2.$$

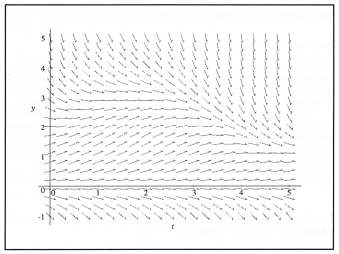
Dividing by the terms on the right hand side and multiplying by $dt$ separates the variables, leaving only the integration:

$$\int \frac{dy}{y(a-by)} = \int dt.$$

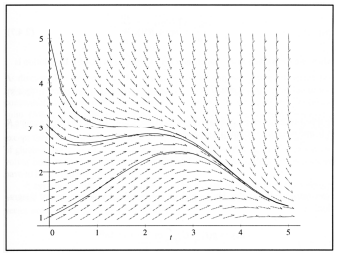
Solution of $\frac{dy}{dt} = y - \frac{y^2}{2+\sin t}$

is [redacted]

Direction field :



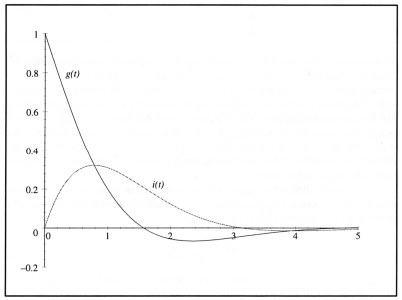
Solutions and direction field :



## Exemple des équations linéaires : glucose et insuline

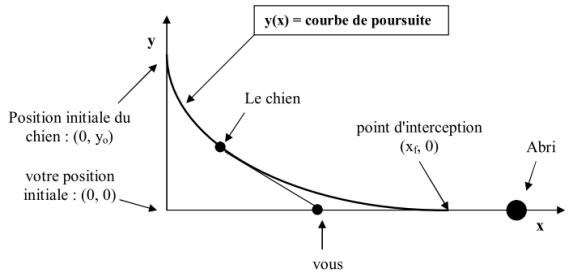
$$\frac{dg}{dt} = -\alpha g - \beta i + p(t)$$
$$\frac{di}{dt} = \gamma g - \delta i.$$

Solutions analytiques :




## Exemple pratique : courbe de poursuite

Vous faites une promenade, marchant le long d'une voie menant droit vers un abri. Soudain, vous apercevez un chien à une certaine distance. Pire, le chien vous aperçoit aussi, et commence à courir vers vous avec l'intention de vous mordre ! Vous commencez à courir vers l'abri, espérant d'y arriver avant que le chien ne vous rattrape. Voici le diagramme de la situation :

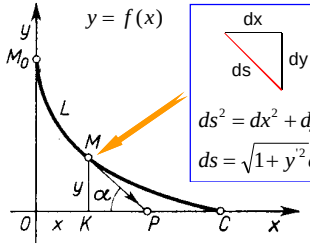



Votre vitesse est ***v***, celle du chien ***u*** (***u > v***). En courant, le chien fixe ses yeux sur vous et se dirige toujours dans votre direction. Trouvez l'équation de la courbe et sa solution. Où exactement va se terminer la poursuite ? Après quel temps ?

**L'équation de la courbe de poursuite**

$$\tan\alpha = -y'$$
$$\tan\alpha = \frac{KM}{KP} = \frac{y}{vt - x}$$

$$y' = \frac{y}{x - vt} \quad \Rightarrow \quad t = \frac{xy' - y}{vy'}$$




$$y = f(x)$$




$$ds^2 = dx^2 + dy^2$$
$$ds = \sqrt{1 + y'^2}\,dx$$

La dérivée de *t* par rapport à *x* donne : $\frac{dt}{dx} = \frac{yy''}{vy'^2}$

D'autre côté, nous avons : $u = \frac{ds}{dt} = \frac{\sqrt{1 + y'^2}\,dx}{dt}$

ce qui donne : $\frac{dt}{dx} = \frac{\sqrt{1 + y'^2}}{u}$

Finalement, par comparaison de et :

$$\frac{yy''}{vy'^2} = \frac{\sqrt{1 + y'^2}}{u} \quad \rightarrow \quad yy'' = ky'^2\sqrt{1 + y'^2} \quad \left(k = \frac{v}{u} < 1\right)$$

Pour résoudre l'équation : $yy'' = ky'^2\sqrt{1 + y'^2}$ $\left(k = \frac{v}{u} < 1\right)$ on substitue : $y' = z$, $z = z(y)$

On obtient : $y\frac{dz}{dy} = kz\sqrt{1 + z^2}$ d'où : $\int \frac{dz}{z\sqrt{1 + z^2}} = \int k\frac{dy}{y}$

Intégration donne : $\ln\left(\frac{1}{z} + \sqrt{1 + \frac{1}{z^2}}\right) = k\ln y + C$ ou $\frac{1}{z} + \sqrt{1 + \frac{1}{z^2}} = C_1 y^k$

La solution par rapport à 1/z = dx/dy :

$$\frac{1}{z} = \frac{1}{2}\left[\left(\frac{y}{y_0}\right)^k - \left(\frac{y}{y_0}\right)^{-k}\right]$$

condition initiale :
$z \to \infty \quad si \quad y \to y_0$

L'intégration finale donne :

$$x = \frac{y_0}{2}\left[\frac{1}{1+k}\left(\frac{y}{y_0}\right)^{1+k} - \frac{1}{1-k}\left(\frac{y}{y_0}\right)^{1-k}\right] + C_2$$

En imposant la condition initiale : $x = 0 \quad si \quad y = y_0$ on obtient :

$$x = \frac{y_0}{2}\left[\frac{1}{1+k}\left(\frac{y}{y_0}\right)^{1+k} - \frac{1}{1-k}\left(\frac{y}{y_0}\right)^{1-k}\right] + \frac{ky_0}{1-k^2}$$

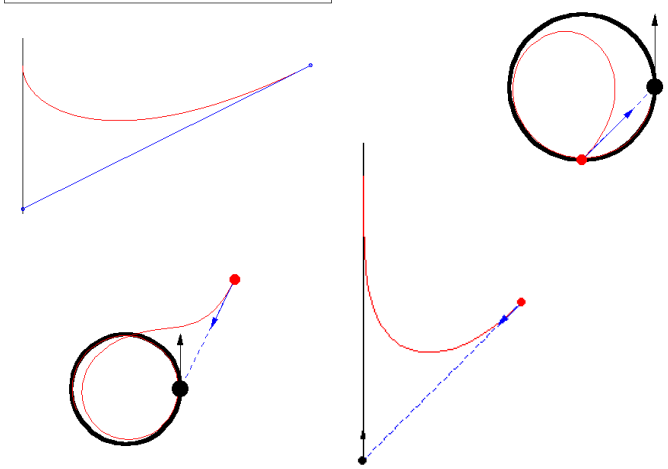
**Où se termine la poursuite ? (il suffit de substituer y=0)**

$$x_f = \frac{ky_0}{1-k^2} = \frac{uvy_0}{u^2 - v^2}$$

**Combien de temps dure la poursuite ?**

$$x_f = vT \quad \Rightarrow \quad T = \frac{x_f}{v} \quad \rightarrow \quad T = \frac{uy_0}{u^2 - v^2}$$

**Exemples de courbes de poursuite :**



**Méthode matricielle de l'intégration d'un jeu d'équations différentielles**

$$\left.\begin{aligned}
\frac{dy_1}{dx} &= a_{11}(x)\cdot y_1 + a_{12}(x)\cdot y_2 + \ldots + a_{1n}(x)\cdot y_n \\
\frac{dy_2}{dx} &= a_{21}(x)\cdot y_1 + a_{22}(x)\cdot y_2 + \ldots + a_{2n}(x)\cdot y_n \\
&\ldots \\
\frac{dy_n}{dx} &= a_{n1}(x)\cdot y_1 + a_{n2}(x)\cdot y_2 + \ldots + a_{nn}(x)\cdot y_n
\end{aligned}\right\} \quad \frac{dy_k}{dx} = \sum_{l=1}^{n} a_{kl}(x)\cdot y_l + b_k \quad (k = 1, 2, \ldots, n)$$

Analogie par rapport à une seule équation :

$$\frac{dY}{dx} = AY + B \qquad Y = \begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ y_n \end{bmatrix}$$

La solution générale du système non-homogène peut être trouvée comme la somme de la solution du système **homogène** et d'une solution particulière du système **non-homogène** :

$$\frac{dY_0}{dx} = AY_0 \qquad \frac{dY_s}{dx} = AY_s + B$$

La preuve est facile : si on ajoute à $\boldsymbol{Y_s}$ une fonction quelconque $\boldsymbol{Z}$ :

$$\frac{dY_s}{dx} + \frac{dZ}{dx} = AY_s + AZ + B$$

$$\frac{dZ}{dx} = AZ$$

$$d'où$$

$$Z \equiv Y_0$$

On transforme le système en forme canonique :

$$\boxed{\frac{d(SY)}{dx} = \frac{dS}{dx}Y + S\frac{dY}{dx}}$$

$$\frac{dY}{dx} = AY + B \quad \longrightarrow \quad S\frac{dY}{dx} = SAY + SB$$

$$\frac{d(SY)}{dx} = \left(SA + \frac{dS}{dx}\right)\left(S^{-1}S\right)Y + SB$$

$$S \neq S(x) \;\Rightarrow\; \frac{dS}{dx} = 0$$

$$\frac{d(SY)}{dx} = \left(SAS^{-1}\right)SY + SB$$

$$\frac{d(SY)}{dx} = D\cdot SY + SB$$

Introduisons deux nouveaux symboles : $\begin{cases} SY = Z \\ SB = E \end{cases}$

pour obtenir la forme canonique des équations : $\dfrac{dZ}{dx} = DZ + E$

Tout d'abord, la solution du système homogène : $\dfrac{dZ_0}{dx} = DZ_0$

$$\frac{dZ_0}{Z_0} = Ddx$$

$$\int \frac{dZ_0}{Z_0} = \int Ddx$$

$$\ln(Z_0) = Dx + C$$

$$Z_0 = C\cdot e^{Dx} \qquad C = \begin{bmatrix} c_1 \\ c_2 \\ \ldots \\ c_n \end{bmatrix}$$

La solution du système **non-homogène** (même forme que le terme libre) :

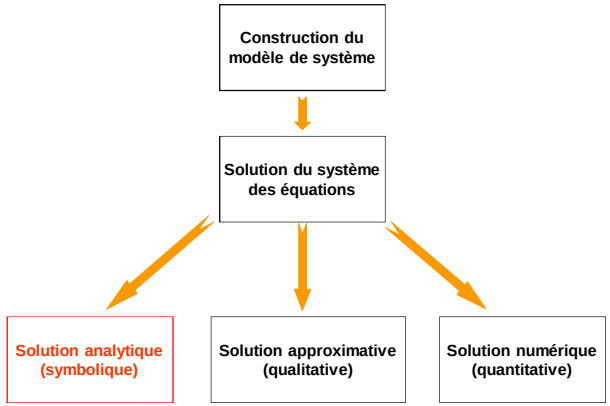
$$Z_s = const$$
$$0 = DZ_s + E$$
$$Z_s = -D^{-1}E$$

D'où la forme générale de la solution :

$$Z = Ce^{Dx} - D^{-1}E$$
$$(D = SAS^{-1}, E = SB)$$

La solution finale du système :

$$\boxed{Y = S^{-1}Z}$$



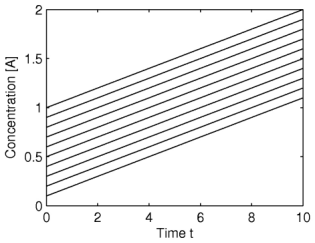

Réactions élémentaires : production constante

$$\xrightarrow{k_1} A \qquad \boxed{\frac{dA}{dt} = k_1}$$

$$dA = k_1 dt$$

$$A = \int dA = \int k_1 dt$$

$$A = k_1 \int dt = k_1 t + A_0$$




(Synthèse constante, e.g. expression génique, sans dégradation mARN/protéine)

Réactions élémentaires : dégradation linéaire

$$A \xrightarrow{k_1} \qquad \boxed{\frac{dA}{dt} = -k_1 A}$$

$$\frac{dA}{A} = -k_1 dt$$

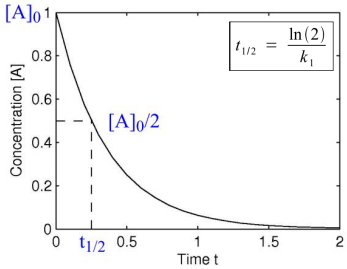
$$\int \frac{dA}{A} = -\int k_1 dt$$

$$\ln(A) = -k_1 \int dt = -k_1 t + C$$

$$A(t) = \exp(-k_1 t + C) = e^{-k_1 t} \cdot e^C$$

$$A(t = 0) = A_0$$

$$A_0 = e^C \longrightarrow A(t) = A_0 e^{-k_1 t}$$



## Réactions élémentaires : autocatalyse

$$\xrightarrow{k_1\cdot[A]} A \qquad \boxed{\frac{dA}{dt} = +k_1 A}$$

$$\frac{dA}{A} = k_1 dt$$

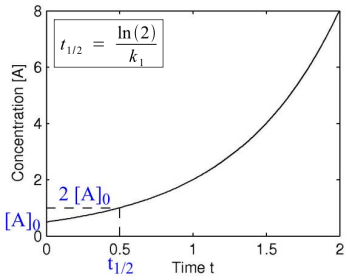
$$\int \frac{dA}{A} = \int k_1 dt$$

$$\ln(A) = k_1 \int dt = k_1 t + C$$

$$A(t) = \exp(k_1 t + C) = e^{k_1 t} \cdot e^{C}$$

$$A(t=0) = A_0$$

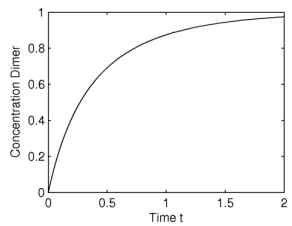
$$A_0 = e^{C} \Rightarrow A(t) = A_0 e^{k_1 t}$$




## Réactions élémentaires : dimérisation

$$A + B \xrightarrow{k_1} AB \qquad \begin{cases} \frac{d[A]}{dt} = -k_1[A]\cdot[B] \\ \frac{d[B]}{dt} = -k_1[A]\cdot[B] \\ \frac{d[AB]}{dt} = k_1[A]\cdot[B] \end{cases}$$

$$[AB] = \frac{A_0 B_0 \left[1 - e^{-(A_0 - B_0)k_1 t}\right]}{A_0 - B_0 e^{-(A_0 - B_0)k_1 t}}$$




## Dimérisation : solution

$$\frac{d[AB]}{dt} = k_1[A]\cdot[B] \xrightarrow[{[B] = B_0 - [AB]}]{[A] = A_0 - [AB]} \boxed{\frac{d[AB]}{dt} = k_1(A_0 - [AB])\cdot(B_0 - [AB])}$$

$$\frac{d[AB]}{(A_0 - [AB])\cdot(B_0 - [AB])} = k_1 dt \Rightarrow \int \frac{d[AB]}{(A_0 - [AB])\cdot(B_0 - [AB])} = \int k_1 dt = k_1 t + C$$

$$\int \frac{d[AB]}{(A_0 - [AB])\cdot(B_0 - [AB])} = \frac{1}{B_0 - A_0}\left[\int \frac{d[AB]}{(A_0 - [AB])} - \int \frac{d[AB]}{(B_0 - [AB])}\right]$$

$$\frac{1}{B_0 - A_0} \ln\left(\frac{B_0 - [AB]}{A_0 - [AB]}\right) = k_1 t + C$$

$$\boxed{[AB] = \frac{B_0 - A_0 \cdot C e^{-(A_0 - B_0)k_1 t}}{1 - C e^{-(A_0 - B_0)k_1 t}}} \qquad (A_0 > B_0)$$

## Solution : conditions initiales

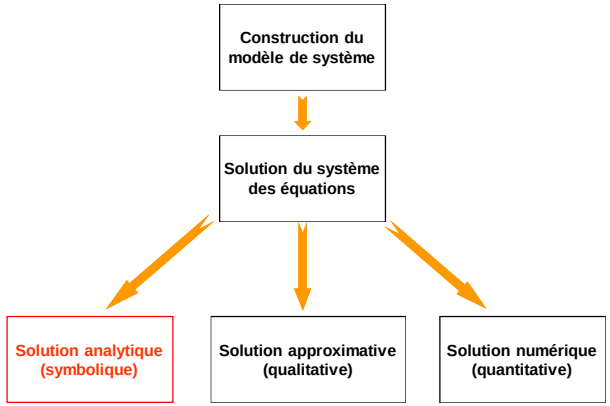
$$[AB] = \frac{B_0 - A_0 \cdot C e^{-(A_0 - B_0)k_1 t}}{1 - C e^{-(A_0 - B_0)k_1 t}}$$

$$[AB]_{t=0} = 0 \Rightarrow 0 = \frac{B_0 - A_0 \cdot C}{1 - C} \Rightarrow C = \frac{B_0}{A_0}$$

$$\boxed{[AB] = \frac{A_0 B_0 \left[1 - e^{-(A_0 - B_0)k_1 t}\right]}{A_0 - B_0 e^{-(A_0 - B_0)k_1 t}}}$$

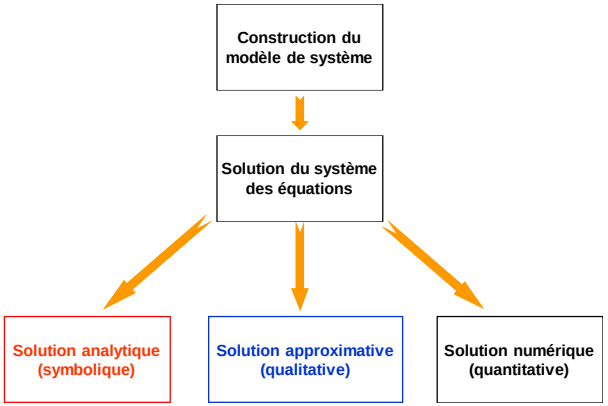
Qu'est-ce qui se passe après un temps très long ?

$$[AB]_{t=\infty} = \frac{A_0 B_0 [1 - 0]}{A_0 - B_0 \cdot 0} = B_0 \qquad (A_0 > B_0)$$




Optimale, prédictive

Rarement disponible




Optimale, prédictive

Rarement disponible

In enzymatic catalysis, a molecule of **S**ubstrate binds to **E**nzyme, forming a **Michaelis complex**. As a result of this interaction a molecule of **P**roduct is released:

$$S + E \underset{k_{-1}}{\overset{k_{+1}}{\rightleftharpoons}} [ES] \xrightarrow{k_{+2}} P + E$$

Complex synthesis is a **second order** reaction while that of its dissociation is a **first order** reaction. Evolution of concentrations of all components can be described by the following set of equations:

$$\begin{cases} \dfrac{dS}{dt} = -k_{+1}SE + k_{-1}[ES] & \text{(a)} \\ \dfrac{dE}{dt} = -k_{+1}SE + k_{-1}[ES] + k_{+2}[ES] & \text{(b)} \\ \dfrac{d[ES]}{dt} = k_{+1}SE - k_{-1}[ES] - k_{+2}[ES] & \text{(c)} \\ \dfrac{dP}{dt} = k_{+2}[ES] & \text{(d)} \end{cases}$$

By adding **(b)** to **(c)** we find that the total enzyme concentration is constant:

$$\frac{dE}{dt} + \frac{d[ES]}{dt} = \frac{d}{dt}(E + [ES]) = 0$$

$$E + [ES] = const = E_0$$

We simplify the equations by substituting ***E = E₀ – [ES]*** into **(c)**:

$$\frac{d[ES]}{dt} = k_{+1}S(E_0 - [ES]) - k_{-1}[ES] - k_{+2}[ES]$$

$$\Downarrow$$

$$\frac{d[ES]}{dt} = -(k_{-1} + k_{+2} + k_{+1}S)[ES] + k_{+1}SE_0$$

Typically, S ≈ P ≈ 1 ÷ 10 mM, while E ≈ 1 ÷ 10 μM. Hence, enzymatic activity (particularly *in vitro*) usually takes place in the limit of constant substrate concentration. The stationary solution of the above equation can be obtained by setting ***d[ES]/dt = 0***:

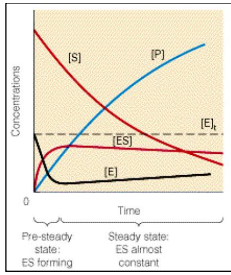
$$[\overline{ES}] = \frac{E_0 S}{K_M + S}$$

$$K_M = \frac{k_{-1} + k_{+2}}{k_{+1}}$$

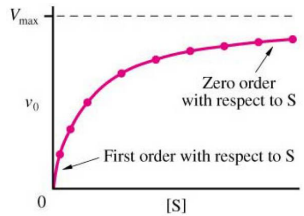
***$K_M$*** is the Michaelis constant, corresponding to the substrate concentration at which half of the enzyme molecules are complexed. The other two equations can be written as:

$$\frac{dS}{dt} = -k_{+1} SE + k_{-1}[\overline{ES}] = -k_{+2}[\overline{ES}]$$

$$\frac{dP}{dt} = k_{+2}[\overline{ES}] = -\frac{dS}{dt} = \frac{k_{+2} SE_0}{K_M + S}$$




The more ES present, the faster ES will dissociate into E + P or E + S. Therefore, when the reaction is started by mixing enzymes and substrates, the [ES] builds up at first, but quickly reaches a STEADY STATE, in which [ES] remains constant. This steady state will persist until almost all of the substrate has been consumed.




For large substrate concentrations the enzymatic reaction rate reaches a constant value, called the ***maximal enzymatic reaction rate***:

$$\frac{dP}{dt} = -\frac{dS}{dt} = \frac{k_{+2} SE_0}{K_M + S} \xrightarrow{S \to \infty} k_{+2} E_0$$

It describes the number of catalytic reactions per unit of time performed by enzyme when it is fully saturated with substrate.

*In reality, reactions may be much more complex than the above equations, and yet it is possible to describe a system accurately with a relatively simple approach. Think of the chemical reaction:*

$$2H_2 + O_2 \rightarrow 2H_2O$$

*Many of the intermediate states are not reflected at all, since only the initial conditions and final results are taken into account. Similarly, in biology it is quite sufficient to monitor the substrate and product concentrations, describing the complex network of intermediate reactions by the simple concept of the enzyme.*

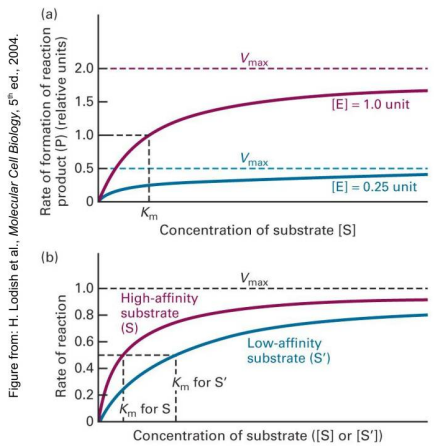
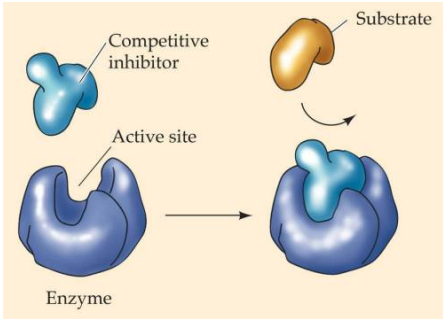



Figure from: H. Lodish et al., *Molecular Cell Biology*, 5th ed., 2004.

$$v = \frac{v_{max}[S]_0}{[S]_0 + K_M}$$

- $v_{max}$: Maximal reaction velocity.
- $K_M$: Affinity enzyme-substrate (subtrate conc. at half-maximal rate).

Inhibition compétitive (isostérique)




$$E + S \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} ES \xrightarrow{k_2} E + P$$

$$EI + S \underset{k_3}{\overset{k_{-3}}{\rightleftharpoons}} E + S + I \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} ES + I \xrightarrow{k_2} E + P + I$$

Steady state: $\frac{d[E]}{dt} = \frac{d[ES]}{dt} = \frac{d[EI]}{dt} = 0$

$$[E]_0 = [E] + [ES] + [EI] \quad \textbf{(1)}$$

$$\frac{d[E]}{dt} = 0 = -k_1[E][S] + k_{-1}[ES] + k_2[ES] - k_3[E][I] + k_{-3}[EI] \quad \textbf{(2)}$$

$$\frac{d[ES]}{dt} = 0 = k_1[E][S] - k_{-1}[ES] - k_2[ES] \quad \textbf{(3)}$$

$$\frac{d[EI]}{dt} = 0 = k_3[E][I] - k_{-3}[EI] \quad \textbf{(4)}$$

From equation (**3**), we can define *E* in terms of *ES* by rearranging to

$$[E] = \frac{(k_{-1} + k_2)[ES]}{k_1[S]} \quad \textbf{(5)}$$

Substituting equation (**5**) into equation (**4**), we have

$$[EI] = \frac{K_m k_3 [I][ES]}{k_{-3}[S]}$$

At this point, we can define the dissociation constant for the inhibitor as $K_i = k_{-3}/k_3$, giving

$$[EI] = \frac{K_m[I][ES]}{K_i[S]} \quad \textbf{(6)}$$

substitute equation (**5**) and equation (**6**) into equation (**1**):

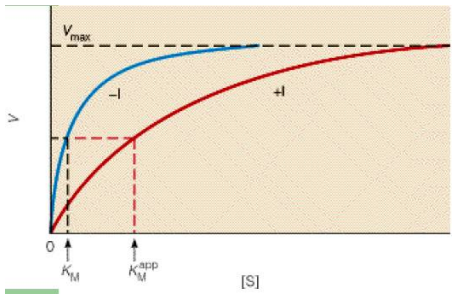
$$[E]_0 = [ES]\left(\frac{K_m}{[S]} + 1 + \frac{K_m[I]}{K_i[S]}\right) = [ES]\frac{K_m K_i + K_i[S] + K_m[I]}{K_i[S]}$$

$$[ES] = \frac{K_i[S][E]_0}{K_m K_i + K_i[S] + K_m[I]} \quad \textbf{(7)}$$

Returning to our expression for $V_0$, we now have:

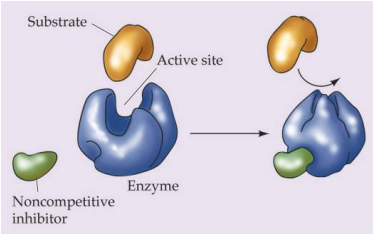
$$V_0 = k_2[ES] = \frac{k_2 K_i[S][E]_0}{K_m K_i + K_i[S] + K_m[I]}$$

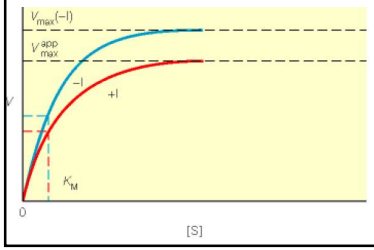
$$= \frac{k_2[E]_0[S]}{K_m + [S] + K_m\frac{[I]}{K_i}} \longrightarrow \boxed{V_0 = \frac{V_{\max}[S]}{K_m(1 + \frac{[I]}{K_i}) + [S]}}$$




Reduced apparent affinity enzyme-substrate

Inhibition non-compétitive (allostérique)




$$\frac{dP}{dt} = k_{+2}\frac{SE_0 / \left(1+\frac{I_0}{K_I}\right)}{K_M + S}$$

Reduced maximal reaction velocity

Construction du modèle de système

Solution du système des équations

Solution analytique (symbolique)

Solution approximative (qualitative)

Solution numérique (quantitative)

Optimale, prédictive — Symbolique

Rarement disponible — Validité limitée

Construction du modèle de système

Solution du système des équations

Solution analytique (symbolique)

Solution approximative (qualitative)

Solution numérique (quantitative)

Optimale, prédictive — Symbolique

Rarement disponible — Validité limitée

**Closed Form Solution for Time-dependent Enzyme Kinetics**

S. Schnell*,† and C. Mendoza‡,§

*J. theor. Biol.* (1997) **187,** 207–212

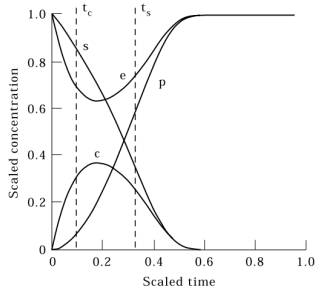



Fig. 1. Time-dependent behaviour of the reactant concentrations in an enzyme–substrate reaction. The infinite time range has been mapped onto the interval (0, 1) with the aid of the exponential time scale $\tau = 1 - 1/\ln(t + e)$, and the reactant concentrations have been non-dimensionally scaled as follows: free substrate $s = [S]/[S_0]$; free enzyme $e = [E]/[E_0]$; complex $c = [ES]/[E_0]$ and product $p = [P]/[S_0]$. The fast, $t_C$, and slow, $t_S$, time-scales are also shown.

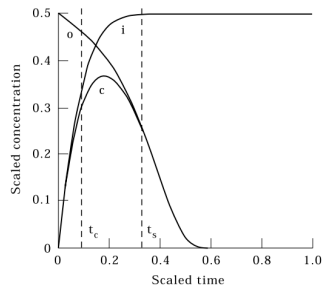
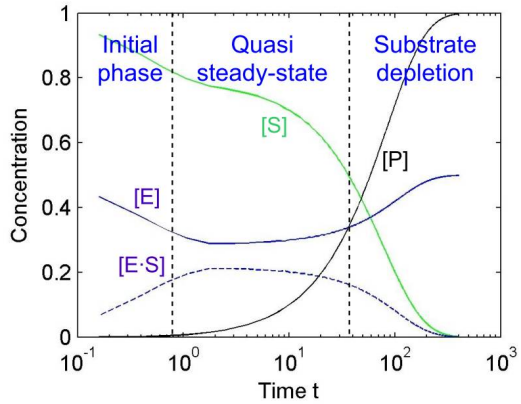



FIG. 3. Comparison of the present closed-form solution for the complex concentration (curve c) with those obtained by singular perturbation methods to order $[E_0]/[S_0]$. It may be appreciated that the agreement with the inner solution (curve i) during the fast transient ($t < t_c$) and with the outer solution (curve o) for $t > t_s$ is very good. However, data from the perturbation methods are noticeably different in the important transition interval $t_c < t < t_s$.

## Enzyme Kinetics: Michaelis-Menten




Intégration numérique des équations différentielles

Equation différentielle typique :

$$\begin{cases} y' = f(x, y(x)) \\ y(x_0) = y_0 \end{cases}$$

Méthode d'Euler :

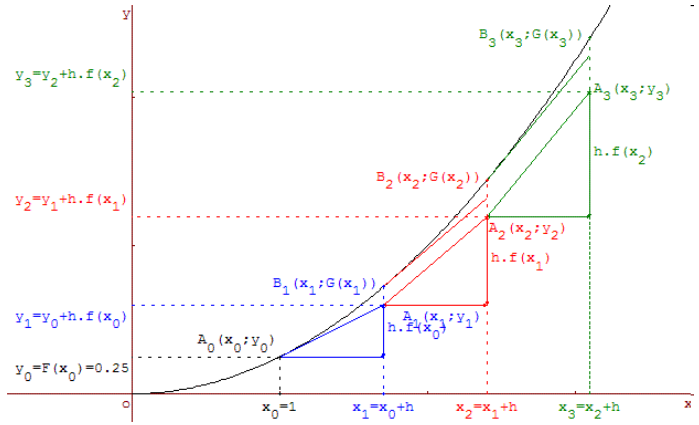
$$y' \approx \frac{y(x+\Delta x) - y(x)}{\Delta x} \quad \Rightarrow \quad y' \cdot \Delta x \approx y(x+\Delta x) - y(x)$$

$$y(x+\Delta x) = y(x) + y' \cdot \Delta x$$

$$\boxed{y(x+\Delta x) = y(x) + f(x, y(x)) \cdot \Delta x}$$

$$y(x_1) = y(x_0) + f(x_0, y(x_0)) \cdot \Delta x$$

$$\boxed{y(x_{k+1}) = y(x_k) + f(x_k, y(x_k)) \cdot \Delta x}$$



$y' = y, y(0) = 1.$

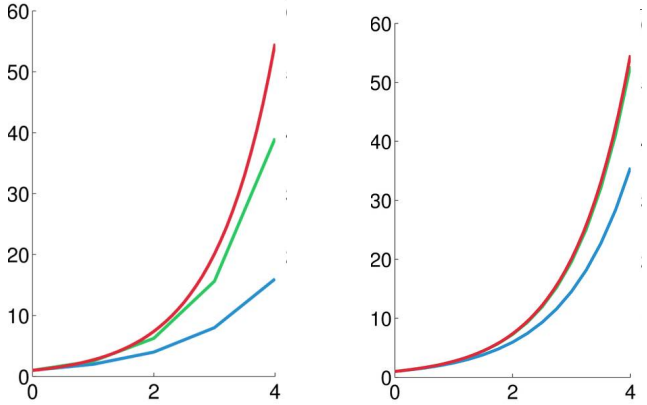
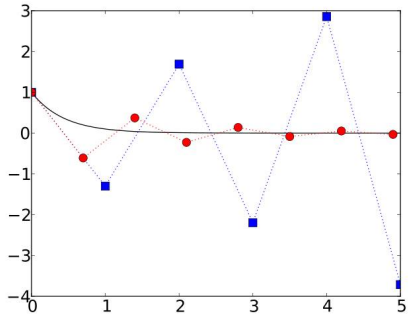


Illustration of several different methods of numerical integration:
**Blue** is the Euler method, **green**, the midpoint method, **red**, the exact solution.
The step size is $h$ = 1.0 (left) and 0.25 (right).

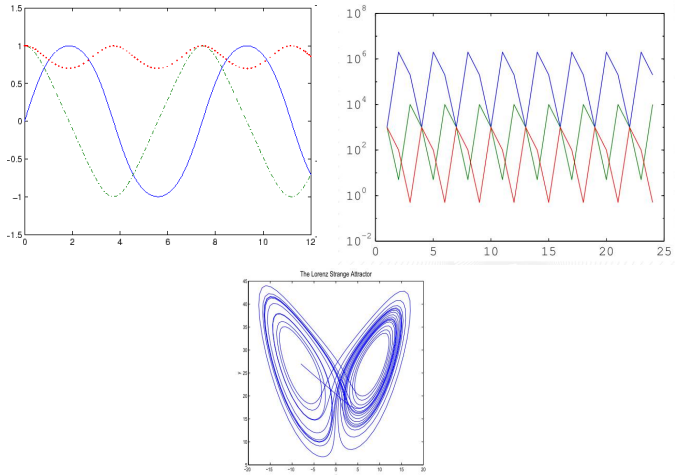
Précision, convergence, stabilité…

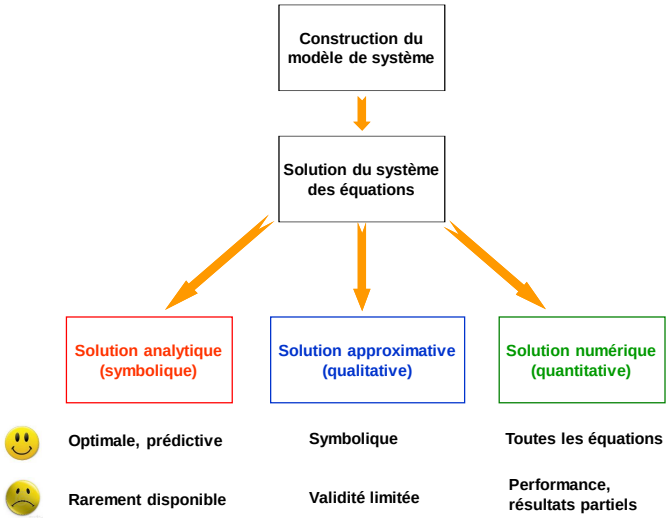
$y' = -2.3y$



Solution computed with the Euler method with step size *h=1* (blue squares) and *h=0.7* (red circles). The black curve shows the exact solution.

Matlab et les solutions numériques



M2I BBS

# Modélisation et Simulation de Systèmes Biologiques

(#2 - mét. num.)

Georges Czaplicki, UPS / IPBS-CNRS
Tél. : 05.61.17.54.04, email : cgeorge@ipbs.fr

## Système positionnel de numération

$$123 = ?$$

$$123 = \square \cdot 10^2 + \square \cdot 10^1 + \square \cdot 10^0$$

$$Valeur = \sum_{i=1}^{n} chiffre \cdot (Base)^{n-i}$$

$$chiffre = \{0, 1, 2, ..., Base - 1\}$$

$$123_5 = 1 \cdot 5^2 + 2 \cdot 5^1 + 3 \cdot 5^0 = 25 + 10 + 3 = 38$$

$$123_8 = 1 \cdot 8^2 + 2 \cdot 8^1 + 3 \cdot 8^0 = 64 + 16 + 3 = 83$$

## Systèmes utiles : binaire, octal, hexadécimal

B=2 : { 0, 1 } | B=8 : {0,1,2,3,4,5,6,7} | B=16 : {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

1 bit :

| | |
|---|---|
| 0 | 0 |
| 1 | 1 |

$2^1=2$

2 bits :

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 2 |
| 1 | 1 | 3 |

$2^2 = 4$

3 bits :

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 2 |
| 0 | 1 | 1 | 3 |
| 1 | 0 | 0 | 4 |
| 1 | 0 | 1 | 5 |
| 1 | 1 | 0 | 6 |
| 1 | 1 | 1 | 7 |

$2^3 = 8$

4 bits :

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 |
| ... | ... | ... | ... | |
| 1 | 0 | 0 | 0 | 4 |
| 1 | 0 | 0 | 1 | 5 |
| ... | ... | ... | ... | |
| 1 | 1 | 1 | 1 | 15 |

$2^4 = 16$

8 bits = 1 octet

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 64 |
| ... | ... | ... | ... | ... | ... | ... | ... | |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 128 |
| ... | ... | ... | ... | ... | ... | ... | ... | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 255 |

$2^8 = 256$

$$123_{10} = 1111011_2 = 173_8 = 7B_{16}$$

## Précision, erreurs d'arrondi, stabilité…

| | sign bit | 8-bit exponent | this bit could be "phantom" | 23-bit mantissa |
|---|---|---|---|---|
| ½ = | 0 | 1 0 0 0 0 0 0 0 | 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 3 = | 0 | 1 0 0 0 0 0 1 0 | 1 | 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| ¼ = | 0 | 0 1 1 1 1 1 1 1 | 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| $10^{-7}$ = | 0 | 1 1 0 1 0 0 1 | 1 | 1 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 0 0 1 0 1 0 |
| = | 0 | 1 0 0 0 0 0 1 0 | | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| $3 + 10^{-7}$ = | 0 | 1 0 0 0 0 0 1 0 | | 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |

$$s \times M \times B^{e-E}$$

1. représentation interne de nombres (entiers, virgule flottante)
2. précision de la machine (liée avec la mantisse)
3. le moins grand nombre représentable (liée avec l'exposant)
4. arrondissage (si cumulable, calcul instable)

IEEE 754

**Exemple :**

| | |
|---|---|
| $r = \sqrt{x^2 + y^2}$ | $x > y \quad \Rightarrow \quad r = \sqrt{x^2\left(1+\left(\frac{y}{x}\right)^2\right)} = x\sqrt{1+\left(\frac{y}{x}\right)^2}$ |
| **pour éviter le dépassement des registres :** | **le gain est donné par :** |
| $x^2 \le M$ <br> $x \le \sqrt{M} = M^{\frac{1}{2}} \xrightarrow{32\,bits} 2^{\frac{32}{2}} = 2^{16}$ | $M = 2^n \quad \Rightarrow \quad n = \frac{\ln M}{\ln 2}$ <br> $M \to \frac{M}{\sqrt{2}} \quad \Rightarrow \quad n = 31.5$ |

# How Java's Floating-Point Hurts Everyone Everywhere

by
Prof. W. Kahan and Joseph D. Darcy
Elect. Eng. & Computer Science
Univ. of Calif. @ Berkeley

Originally presented 1 March 1998
at the invitation of the
ACM 1998 Workshop on Java for
High–Performance Network Computing
held at Stanford University

http://www.cs.ucsb.edu/conferences/java98

This document: http://www.cs.berkeley.edu/~wkahan/JAVAhurt.pdf or
http://www.cs.berkeley.edu/~darcy/JAVAhurt.pdf

## L'explosion de la fusée Ariane 5 (le 4 juin 1996)

- Après 37 secondes de vol, les fortes accélérations de la fusée provoquent un dépassement de capacité dans le calculateur du système de guidage inertiel principal, qui se met aussitôt hors service.
- Le système de guidage de secours (identique à l'autre) subit la même avarie, et s'arrête à la même seconde.
- Le pilote automatique se met en route. Suite à une mauvaise interprétation du signal de panne des deux guidages inertiels hors service, le pilote automatique provoque une violente correction de trajectoire. La fusée dérape de sa trajectoire, et les boosters sont arrachés par le courant d'air décentré. Ce qui déclenche le mécanisme d'autodestruction préventive de la fusée.

## L'échec du missile *Patriot* (le 25/02/1991, guerre du Golfe)

Un missile américain *Patriot* à Dharan (Arabie Saoudite) a échoué de détruire un missile iraquien *Scud*. Ce dernier a frappé les casernes de l'armée américaine, tuant 28 soldats et blessant ~100 personnes.

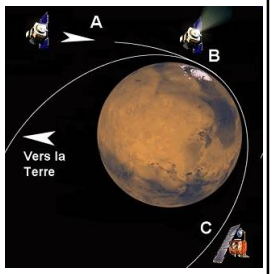
Les résultats d'enquête ont été publiés dans le rapport *Patriot Missile Defense: Software Problem Led to System Failure at Dhahran, Saudi Arabia*. Il s'avère que le problème venait du calcul de temps écoulé depuis le démarrage du système, qui montrait les erreurs d'arrondi.

Concrètement, l'horloge interne mesurait le temps en dixièmes de secondes, ce qui était multiplié par 10 pour arriver aux secondes. Le calcul se faisait sur des registres de 24 bits. Puisque la valeur 1/10 en système binaire va au-delà de 24 bits, ce qui dépassait était coupé. La petite erreur d'arrondi, multipliée par des grandes chiffres représentant le temps en 1/10 s, conduisait de façon cumulative à une grande erreur finale. L'erreur d'arrondi sur chaque mesure était d'ordre de 0.1µs, mais sur 100h après le démarrage, ceci donne $0.095 \cdot 10^{-6} \cdot 100 \cdot 60 \cdot 60 \cdot 10 = 0.342$ s.

Un missile *Scud* a la vitesse de 1676 m/s, ce qui veut dire qu'il parcourt presque 600 m pendant ce temps (0.342 s). Par conséquent, il était au-dehors du *tracking range* (champ de poursuite) du missile *Patriot*.

## Perte de la sonde spatiale *Mars Climate Orbiter* (le 23/09/1999)

Le 23/09/1999 débutait la mise en orbite de la sonde *Mars Climate Orbiter*. Pour placer la sonde en orbite martienne, il est nécessaire de la ralentir suffisamment pour qu'elle puisse être happée par le champ de gravité de Mars. Si le freinage n'est pas assez fort, la sonde survole Mars puis dépasse la planète en continuant sur sa lancée. Le moteur devait fonctionner pendant ~16 minutes pour assurer un freinage suffisant. A 11:01, alors que la sonde est en train de frôler le pôle nord martien, le moteur de 640 Newtons de poussée est mis à feu. La sonde devait réapparaître de l'autre côté de Mars à 11:26:25, mais l'antenne n'a capté aucun signal. A 11:41, la NASA annonce officiellement que le contact avec la sonde est perdu.




Il semble que la perte de *Mars Climate Orbiter* doit être mise sur le compte d'un problème d'unité dans l'expression d'une force de poussée. Les ingénieurs de *Lockheed Martin Astronautics* (Denver, Colorado), la firme qui a conçu et fabriqué la sonde martienne, avaient apparemment gardé l'habitude de travailler avec les unités du système anglo-saxons (poussée du moteur en livres). De leur côté, les ingénieurs du *Jet Propulsion Laboratory* (Pasadena, Californie) travaillaient depuis des années dans le système métrique (poussée calculée en Newtons ; une livre équivaut à 4,48 Newtons). lors du transfert des données entre le centre de Lockheed et celui du JPL, personne ne se soit rendu compte qu'il fallait convertir les données, chacun étant persuadé que l'un utilisait les mêmes unités que l'autre !

## Calcul numérique (dérivées, intégrales, équations différentielles,…)

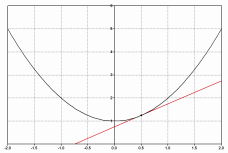
**Développements de Taylor :**

$$f(x+h) = f(x) + \frac{df}{dx}h + \frac{1}{2}\frac{d^2 f}{dx^2}h^2 + O(h^3)$$

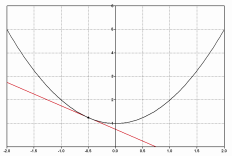
$$f(x-h) = f(x) - \frac{df}{dx}h + \frac{1}{2}\frac{d^2 f}{dx^2}h^2 + O(h^3)$$

**Schémas numériques (avant, arrière, centré) :**

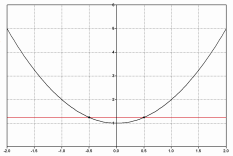
$$f'(x) \approx \frac{f(x+h)-f(x)}{h} + O(h)$$



$$f'(x) \approx \frac{f(x)-f(x-h)}{h} + O(h)$$



$$f'(x) \approx \frac{f(x+h)-f(x-h)}{2h} + O(h^2)$$

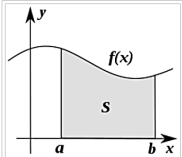


**Calcul direct de dérivées :**

$$f'(x_k) \approx \frac{f(x_{k+1}) - f(x_{k-1})}{x_{k+1} - x_{k-1}}$$

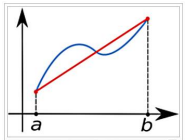
h grand => calcul peu précis
h petit => erreurs d'arrondi

**Intégration numérique : comment trouver $\int_a^b f(x)dx$ ?**




Numerical integration consists of finding numerical approximations for the value $S$

### Méthode de trapèzes




The function $f(x)$ (in blue) is approximated by a linear function (in red).

**Intervalle unique :**

$$\int_a^b f(x)dx \approx (b-a)\frac{f(a)+f(b)}{2}$$

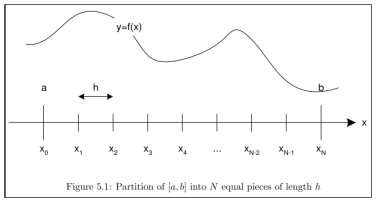



Figure 5.1: Partition of $[a, b]$ into $N$ equal pieces of length $h$

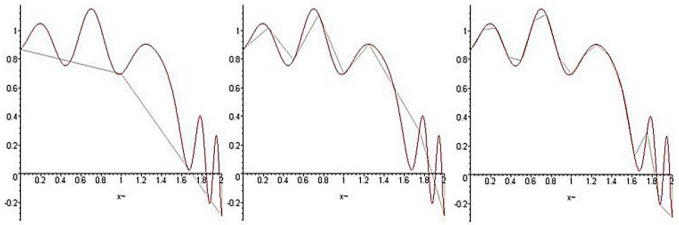
**Intervalles multiples :**

$$\int_a^b f(x)dx \approx \frac{b-a}{n}\left[\frac{f(a)+f(b)}{2} + \sum_{k=1}^{n-1} f\left(a + k\frac{b-a}{n}\right)\right]$$

## Exemple d'intégration numérique

$$= 1.1 + \ln(e - \frac{x}{100} + \frac{3}{5}\tanh(\ln(x+10^{-7})+1))\cos x + \frac{2}{5}(x - \frac{\cos(3x)}{5})^2$$
$$\cdots + \frac{11}{100}\sqrt{2+2x}\sin(\frac{44}{25}(4+3\sqrt{x})x - \frac{19}{20}x^5) - e^{\frac{\pi}{3}}$$

Découpage pour différentes valeurs de *n* (2,8 et 16).



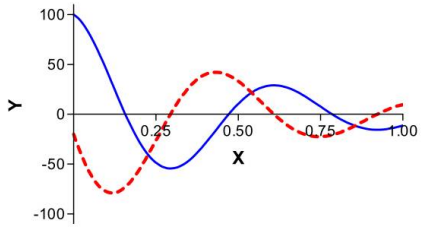
## Résolution des équations numériques

La dérivée possède les propriétés suivantes :

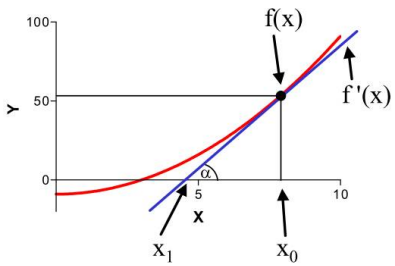
$$f'(x) = \frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

| fonction | dérivée |
|---|---|
| croissante | positive |
| décroissante | négative |
| maximum ou minimum | zéro |

**fonction et sa dérivée**




## La recherche du passage par 0 : méthode de Newton

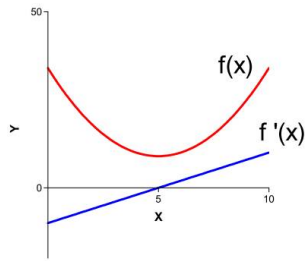



$$f'(x_0) = \tan\alpha = \frac{f(x_0) - 0}{x_0 - x_1}$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

La méthode de Newton peut être utilisée pour trouver le minimum d'une fonction :

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$




$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}$$

## Application de la méthode de Newton aux algorithmes numériques

**Exemple :** comment calculer la racine carrée de A ?
**Solution :** Créer une fonction dont le passage par zéro donne la solution du problème…

$$x = \sqrt{A}$$
$$x - \sqrt{A} = 0$$

…sous la condition que la fonction soit correctement créée !

$$x = \sqrt{A}$$
$$x^2 = A$$
$$x^2 - A = 0$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^2 - A}{2x_i} = \frac{x_i}{2} + \frac{A}{2x_i}$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^2 - A}{2x_i} = \frac{x_i}{2} + \frac{A}{2x_i}$$

**Résultat du calcul pour A=9 et A=16 :**

| | |
|---|---|
| 4.500000000000000 | 8.000000000000000 |
| 3.250000000000000 | 5.000000000000000 |
| 3.009615384615385 | 4.100000000000000 |
| 3.000015360039322 | 4.001219512195122 |
| 3.000000000039321 | 4.000000185844589 |
| 3.000000000000000 | 4.000000000000004 |
| 3.000000000000000 | 4.000000000000000 |

## Intégration numérique des équations différentielles

Méthode d'Euler :

$$\begin{cases} y' = f(x, y(x)) \\ y(x_0) = y_0 \end{cases}$$

$$y(x_{k+1}) = y(x_k) + h \cdot f(x_k, y(x_k))$$

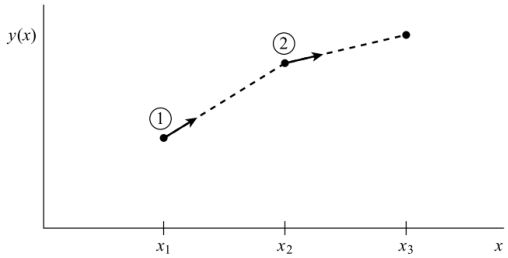



Figure 16.1.1. Euler's method. In this simplest (and least accurate) method for integrating an ODE, the derivative at the starting point of each interval is extrapolated to find the next function value. The method has first-order accuracy.

Méthode du point médian :

$$\begin{cases} k_1 = hf(x_n, y_n) \\ k_2 = hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1\right) \\ y_{n+1} = y_n + k_2 + O(h^3) \end{cases}$$

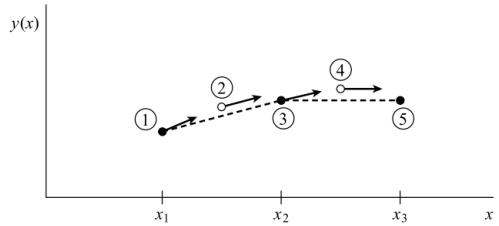



Figure 16.1.2. Midpoint method. Second-order accuracy is obtained by using the initial derivative at each step to find a point halfway across the interval, then using the midpoint derivative across the full width of the interval. In the figure, filled dots represent final function values, while open dots represent function values that are discarded once their derivatives have been calculated and used.

Méthode Runge-Kutta de l'ordre 4 :

$$\begin{cases} k_1 = hf(x_n, y_n) \\ k_2 = hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\ k_3 = hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\ k_4 = hf(x_n + h, y_n + k_3) \\ y_{n+1} = y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5) \end{cases}$$

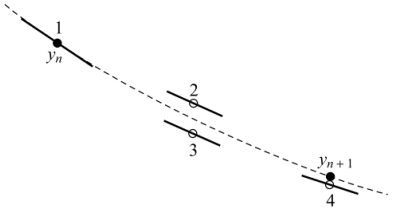
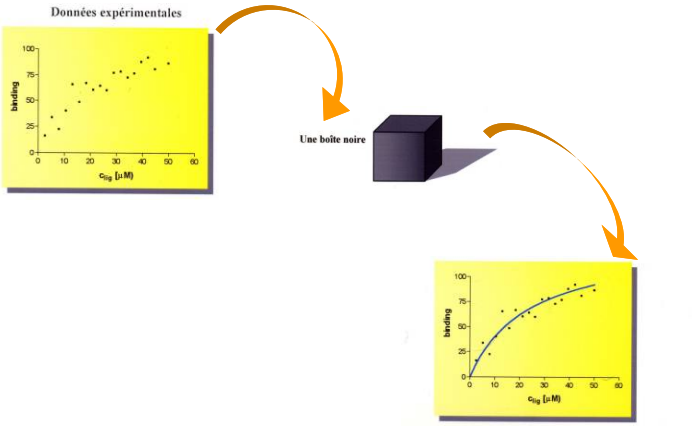


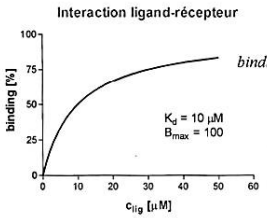

Figure 16.1.3. Fourth-order Runge-Kutta method. In each step the derivative is evaluated four times: once at the initial point, twice at trial midpoints, and once at a trial endpoint. From these derivatives the final function value (shown as a filled dot) is calculated. (See text for details.)

**Techniques d'optimisation**



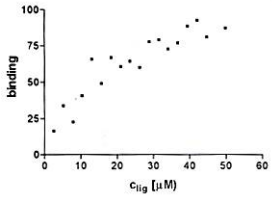

Les résultats du traitement ⇒ paramètres du modèle

A) Modèle d'un phénomène biochimique ou biologique

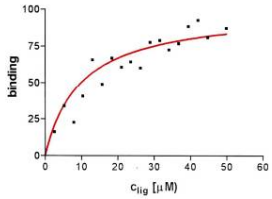



$$binding = \frac{B_{max} \cdot c_{lig}}{K_d + c_{lig}}$$

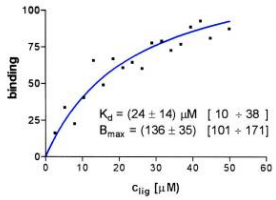
B) Résultats d'une expérience




C) Superposition de la courbe théorique et des points expérimentaux

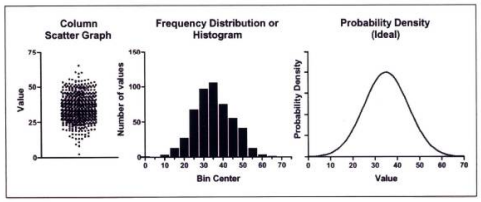



D) Résultat du *fitting*



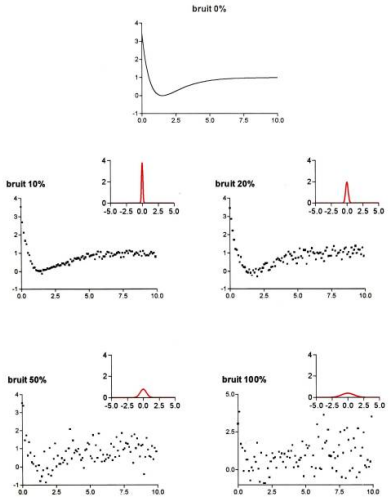
## Distribution gaussienne

Quand un grand nombre de facteurs aléatoires et indépendants s'accumulent, les données apparaissent sous la forme de la distribution dite normale, ou gaussienne :



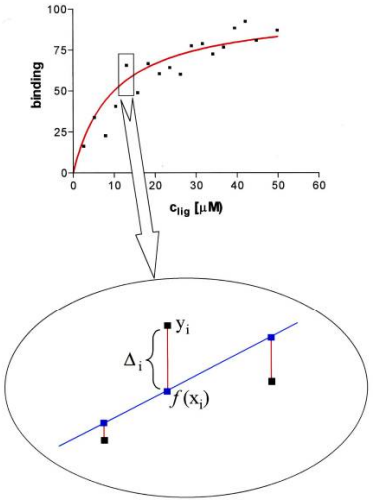

| Distribution d'une grande quantité de données | Distribution de la fréquence d'apparition de points (histogramme) | Distribution gaussienne idéale |
|---|---|---|

## Influence du bruit sur les résultats de mesures

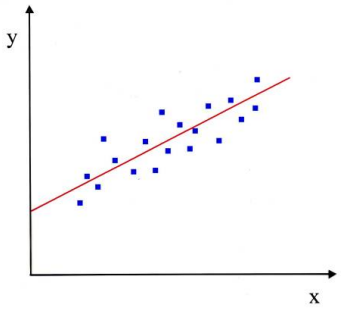



**L'enjeu :**

**Trouver le minimum de la somme des déviations entre les points mesurés et ceux calculés d'après un modèle.**




## Exemple : ligne droite



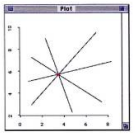

$$y_i = a \cdot x_i + b + \sigma_i$$

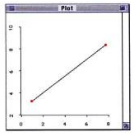
$$y_i = \alpha \cdot x_i + \beta$$

**a, b** : les paramètres du modèle
**α, β** : les estimateurs des paramètres **a, b**

## Nombre de degrés de liberté
**(degrees of freedom)**

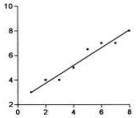
Le nombre de degrés de liberté (DL) représente "**le nombre d'observations moins le nombre de relations parmi ces observations**" (Walker, 1940).





Il est impossible de dessiner une droite de régression si on dispose seulement d'un point de mesure (c.à.d. une observation). En effet, beaucoup de droites peuvent passer par ce point. Il y a une ambiguïté. (**DL = 1 – 2 = -1**).

Pour deux observations, il y a une droite unique qui passe par les deux points et la corrélation est parfaite (r = 1). Mais le bruit contenu dans les mesures détermine le résultat ! La valeur **DL = 2 – 2 = 0**, donc on n'a **aucune liberté** dans l'estimation de paramètres de la droite.



Le nombre de degrés de liberté spécifie le nombre de données utiles pour estimation. **La confiance en résultat s'augmente avec la valeur de DL.**

## Régression linéaire

$$d = \frac{1}{\sigma^2}\sum_{i=1}^{N}\left[y_i - f(x_i)\right]^2 = \min$$

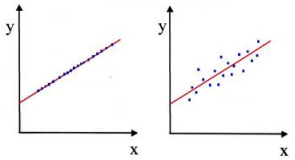
$$f(x) = \sum_{j=1}^{n} a_j \Phi_j(x)$$

$$f(x) = a_1 \cdot \Phi_1(x) + a_2 \cdot \Phi_2(x) + \ldots$$

$$a_1 = a \qquad \Phi_1(x) = x$$
$$a_2 = b \qquad \Phi_2(x) = 1$$
$$f(x) = ax + b$$

$$a_1 = a \qquad \Phi_1(x) = \sin(x)$$
$$a_2 = b \qquad \Phi_2(x) = e^{-x}$$
$$a_3 = c \qquad \Phi_3(x) = x^4$$
$$f(x) = a \cdot \sin(x) + b \cdot e^{-x} + c \cdot x^4$$

$$a = \frac{\langle xy\rangle - \langle x\rangle \cdot \langle y\rangle}{\langle x^2\rangle - \langle x\rangle^2}$$
$$b = \langle y\rangle - a \cdot \langle x\rangle$$



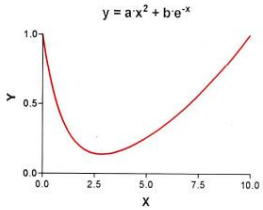

r=0.999 r=0.750

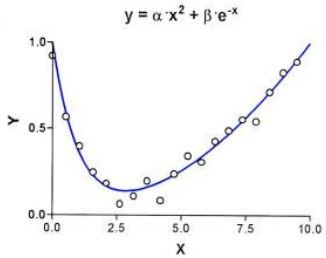
$$r = \frac{\sum_{i=1}^{N}\left(x_i - \langle x\rangle\right)\left(y_i - \langle y\rangle\right)}{\sqrt{\left(\sum_{i=1}^{N}\left(x_i - \langle x\rangle\right)^2\right)\left(\sum_{i=1}^{N}\left(y_i - \langle y\rangle\right)^2\right)}}$$

$$r = \frac{\langle xy\rangle - \langle x\rangle \cdot \langle y\rangle}{\sqrt{\left(\langle x^2\rangle - \langle x\rangle^2\right)\left(\langle y^2\rangle - \langle y\rangle^2\right)}} = a \cdot \sqrt{\frac{\langle x^2\rangle - \langle x\rangle^2}{\langle y^2\rangle - \langle y\rangle^2}}$$

## Régression linéaire

**(approche linéaire par rapport aux paramètres cherchés, courbe non-linéaire par rapport à la variable *x*)**



## Limitations de la régression linéaire

- $f(x) = a \cdot \sin(x) + b \cdot e^{-x} + c \cdot \sqrt{x}$
- $g(x) = \sin(a \cdot x) + e^{-b \cdot x} + \sqrt{x - c}$

- $y = e^{-b \cdot x}$
  $\ln(y) = -b \cdot x$
- $Y = -b \cdot x$

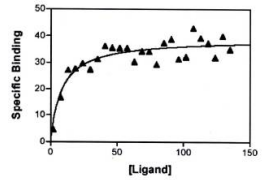
$$y = e^{-a \cdot x} + e^{-b \cdot x}$$

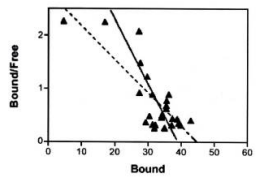
???

**Cas linéaire : calcul direct**

**Cas nonlinéaire : recherche itérative**

## Exemple d'une transformation nonlinéaire




résultat d'ajustement (fitting)

résultat de la régression linéaire sur données transformées

## Cas nonlinéaire (la recherche itérative)

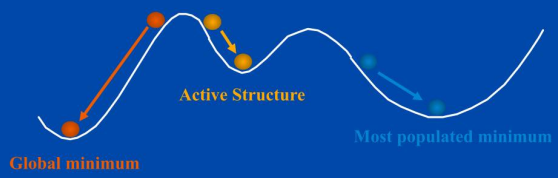
**Problème** :

Le minimum trouvé peut être local plutôt que global


**Solution :**

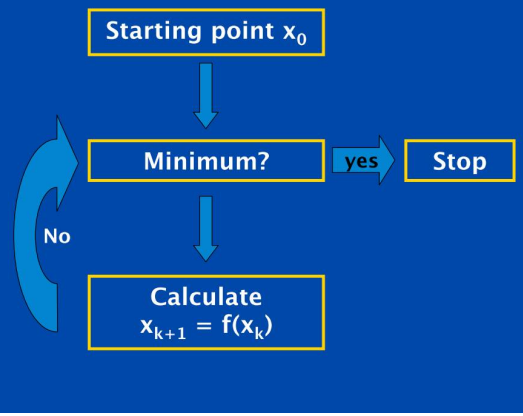
répéter la recherche plusieurs fois avec des points de départ différentes et choisir la solution avec les moindres carrés

## Finding Local and Global Minima




- **Most minimization method can only go downhill and so locate the closest (downhill sense) minimum.**
- **No minimization method can guarantee the location of the global energy minimum.**
- **No method has proven the best for all problems: *try various methods and starting conformation to search for global the minimum***

## A General Iterative Minimization Scheme




## Minimization techniques: line search

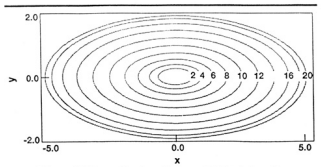


**Figure 4–1. Energy Contour Surface of a Simple Function**

An energy contour surface for the function $x^2 + 5y^2$. Each contour represents an increase of two arbitrary energy units.

$$\{x_i'\} = \left\{ x_{i0} + \alpha \left( \frac{\partial V}{\partial x_i} \right)_0 \right\}$$

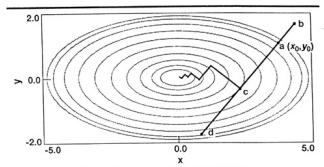


**Figure 4–2. Energy Surface for Eq. 4–1**

The derivative vector from the initial point **a** $(x_0, y_0)$ defines the line search direction. Note that the derivative vector does not point directly toward the minimum. Compare this representation with that in Figure 4–3, where the line (**b–a–c–d**) is searched in one dimension for the minimum. Note that the minimum (point **c**) occurs precisely at the point where the derivative vector is tangent to the energy contours, which implies that the subsequent derivative vectors are orthogonal to the previous derivatives.

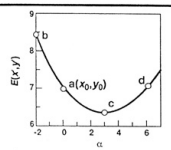


**Figure 4–3. Cross Section of the Energy Surface as Defined by the Intersection of the Line Search Path in Figure 4–2 with the Energy Surface**

The independent variable α is a one-dimensional parameter that is adjusted so as to minimize the value of the function $E(x', y')$, where $x'$ and $y'$ are parameterized in terms of α in Eq. 4–3. Point **a** corresponds to the initial point (when α is 0), and point **c** is the local one-dimensional minimum. Points **b** and **d**, along with **a**, bound the minimum and form the basis for an iterative search for the minimum.

## Steepest descent

$$x_{i+1} = x_i - \varepsilon_i \cdot \nabla V_i$$

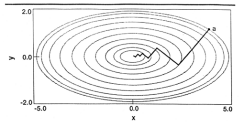


**Figure 4–4. Minimization Path following a Steepest-Descents Path**

When complete line searches starting from point **a** are used, the minimum is reached in about 12 iterations. Here, where a rigorous line search is carried out, approximately 8 function evaluations are needed for each line search using a quadratic interpolation scheme. Note how steepest descents consistently overshoots the best path to the minimum, resulting in an inefficient, oscillating trajectory.

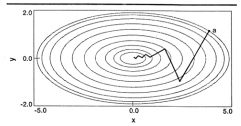
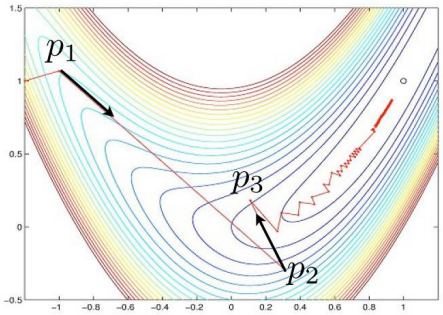


**Figure 4–5. Minimization Path following a Steepest-Descents Path without Line Searches**

The searching starts from point **a** and converges on the minimum in about 12 iterations. Although the number of iterations is slightly larger than in Figure 4–4, the total minimization is five times faster since, on average, each iteration used only 1.3 function evaluations. Note that, in most applications in molecular mechanics, the function evaluation is the most time-consuming portion of the calculation.

**Good** point: stable and initially very fast

**Bad** points: very slow at the final stage, slow convergence for some types of functions




## Conjugate gradients

$$\upsilon_k = -g_k + \gamma_k \upsilon_{k-1}$$

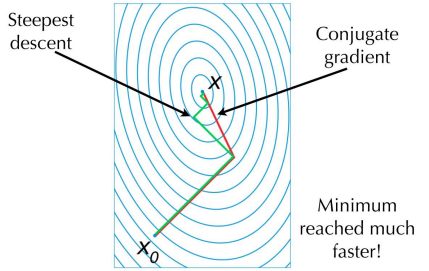
$$\gamma_k = \frac{g_k \cdot g_k}{g_{k-1} \cdot g_{k-1}}$$

**Memory from previous step**

**Broyden, Fletcher, Goldfarb, Shanno**

**Good points:**
- **fixes the slow convergence problem**
- **uses variable search directions instead of gradients only**

**Bad points:**
- **works on quadratic functions only, needs periodic restarts for other functions**



## Methods based on second derivative

**Newton-Raphson algorithm:** $\tan(\phi) = f''(x_0) = \dfrac{f'(x_0)}{x_0 - x_1}$ $\qquad x_1 = x_0 - \dfrac{f'(x_0)}{f''(x_0)}$

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \mathbf{F}^{-1}(\mathbf{x}_i)\cdot\nabla V_i$$

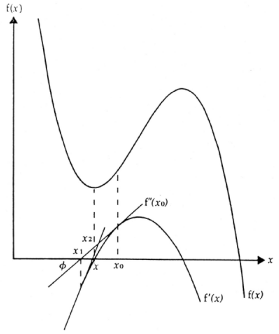



*Figure 3.2. Minimization of a function f(x) by the Newton–Raphson procedure. Iteration starting at $x_0$ improves the solution of $f'(x) = 0$ successively to $x_1$ and to $x_2$, approaching the true solution x.*

$$\left[\begin{array}{ccc|ccc|ccc}
\frac{\delta^2 V}{\delta x_1^2} & \frac{\delta^2 V}{\delta y_1 \delta x_1} & \frac{\delta^2 V}{\delta z_1 \delta x_1} & — & — & — & — & — & — \\
\frac{\delta^2 V}{\delta x_1 \delta y_1} & \frac{\delta^2 V}{\delta y_1^2} & \frac{\delta^2 V}{\delta z_1 \delta y_1} & — & — & — & — & — & — \\
\frac{\delta^2 V}{\delta x_1 \delta z_1} & \frac{\delta^2 V}{\delta y_1 \delta z_1} & \frac{\delta^2 V}{\delta z_1^2} & — & — & — & — & — & — \\
\hline
\frac{\delta^2 V}{\delta x_1 \delta x_2} & \frac{\delta^2 V}{\delta y_1 \delta x_2} & \frac{\delta^2 V}{\delta z_1 \delta x_2} & \frac{\delta^2 V}{\delta x_2^2} & \frac{\delta^2 V}{\delta y_2 \delta x_2} & \frac{\delta^2 V}{\delta z_2 \delta x_2} & — & — & — \\
\frac{\delta^2 V}{\delta x_1 \delta y_2} & \frac{\delta^2 V}{\delta y_1 \delta y_2} & \frac{\delta^2 V}{\delta z_1 \delta y_2} & \frac{\delta^2 V}{\delta x_2 \delta y_2} & \frac{\delta^2 V}{\delta y_2^2} & \frac{\delta^2 V}{\delta z_2 \delta y_2} & — & — & — \\
\frac{\delta^2 V}{\delta x_1 \delta z_2} & \frac{\delta^2 V}{\delta y_1 \delta z_2} & \frac{\delta^2 V}{\delta z_1 \delta z_2} & \frac{\delta^2 V}{\delta x_2 \delta z_2} & \frac{\delta^2 V}{\delta y_2 \delta z_2} & \frac{\delta^2 V}{\delta z_2^2} & — & — & — \\
\hline
— & — & — & — & — & — & & & \\
— & — & — & — & — & — & & \text{etc.} & \\
— & — & — & — & — & — & & &
\end{array}\right]$$

*Figure 3.3. Elements of the block-diagonal and full F matrix. The block-diagonal contains only 3 × 3 blocks where x, y and z belong to the same atom (top left and lower right). The full matrix also contains all other blocks.*

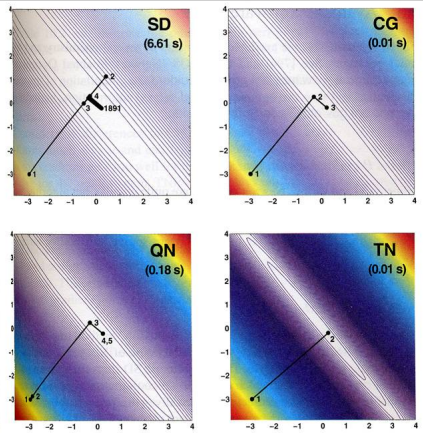



Figure 10.10. Minimization paths (with corresponding CPU times) for a function of two variables shown on top of function contour plots, for steepest descent (SD), nonlinear conjugate gradient (CG), BFGS quasi-Newton (QN), and truncated-Newton (TN) algorithms. See text for functional construction details. All contours are the same, but different levels of resolution are used in each plot to discern both the region near the minimum (darkest contour plot) and the higher-energy regions (lighter plots).

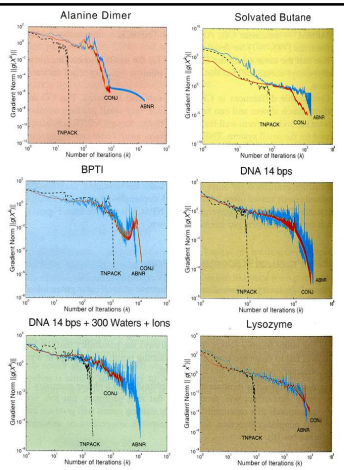
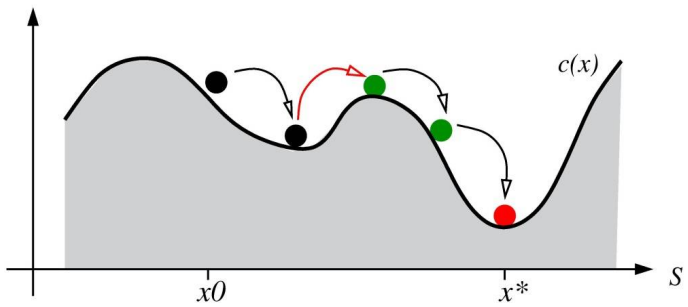



Figure 10.11. Minimization progress (gradient norm) of three CHARMM algorithms (CONJ: nonlinear conjugate gradient, ABNR: adopted-basis Newton-Raphson, and TNPACK: truncated Newton) for various molecular systems. See Figures 10.4 and 10.5 for corresponding sparse matrix patterns; the local Hessians are used as preconditioners for TNPACK. See also Table 10.2 for final function and gradient-norm values and required CPU time.

## Le problème majeur lié avec toutes les techniques de minimisation basées sur le calcul de gradients




**On descend toujours vers le minimum le plus proche du point de départ !**

# Non-Gradient-based, stochastic Monte Carlo

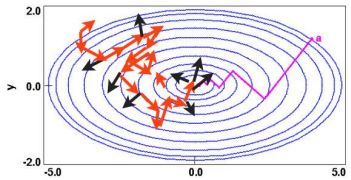
- Calculate a random move, then accept or reject it based on the new energy value and temperature T. ***ran*** is a random number between 0 and 1

v = random vector

if (ran < exp(-(E(x+v)-$E_{old}$)/T)) { then x' = x + v ; otherwise x' = x }

$E_{old}$ = E(x')

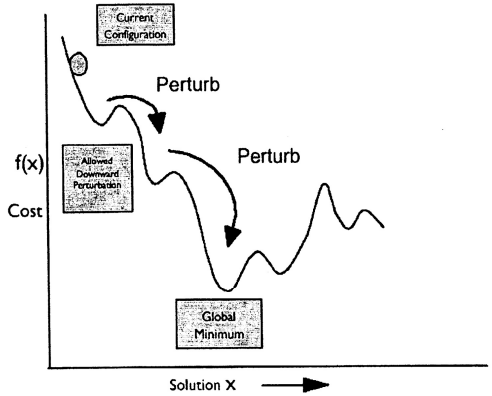
Red arrows are accepted moves. Black arrows rejected moves.




MC always allows a downhill step, and sometimes allows an uphill step. Tha allowance depends on the temperature.
MC can cross barriers.
Best used with simulated annealing.



## Algorithmes stochastiques




## Stochastic algorithms: simulated annealing

$$x_i' = x_i + r \cdot v_i \qquad p = e^{-\frac{f'-f}{T}}$$

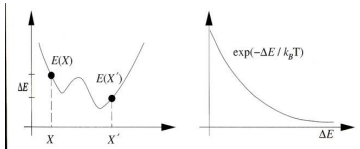
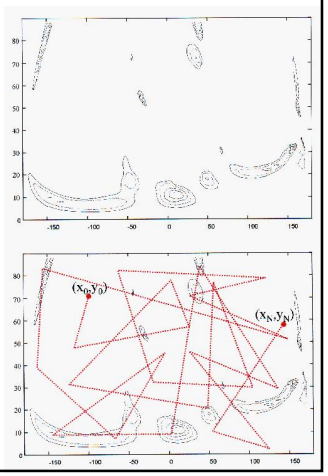


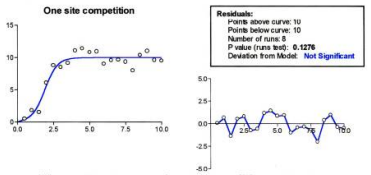

Figure 11.4. The Boltzmann probabilities for two system states $X$ and $X'$ with energies $E(X)$ and $E(X')$ are related at equilibrium by the probability ratio $\exp[-\Delta E/k_B T]$.

**If p > p', accept the new point (p' generated randomly)**

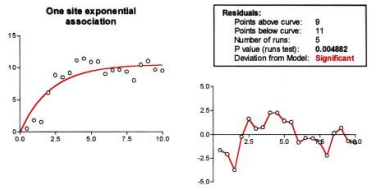
$$T' = r_T \cdot T$$



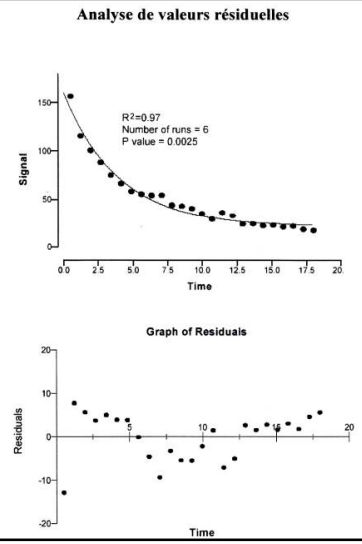

## Valeurs résiduelles




$$y_i^{exp} = f(x_i) + \sigma_i \qquad y_i^{exp} - f(x_i) = \sigma_i$$




$$y_i^{exp} = g(x_i) + \sigma_i \qquad f(x_i) - g(x_i) \overset{?}{=} \sigma_i$$



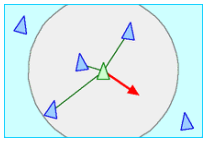

## PSO (Particle Swarm Optimization)

## Introduction to the PSO: **Origins**

- Inspired by the nature social behavior and dynamic movements with communications of insects, birds and fish

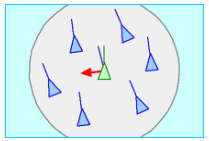
## Introduction to the PSO: **Origins**

- In 1986, Craig Reynolds described this process in 3 simple behaviors:



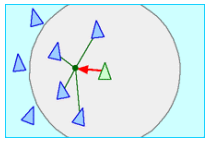
**Separation**

avoid crowding local flockmates



**Alignment**

move towards the average heading of local flockmates

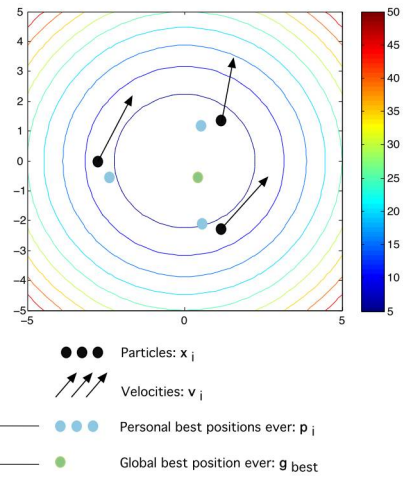


**Cohesion**

move toward the average position of local flockmates

## Terminology

- Particles
- Velocities
- Personal best
- Global best




## Theory - Equations

- The Basic PSO algorithm consists of the velocity:

$$v_i(k+1) = v_i(k) + \gamma_{1i}\left(p_i - x_i(k)\right) + \gamma_{2i}\left(G - x_i(k)\right)$$

- ..and position:

$$x_i(k+1) = x_i(k) + v_i(k+1)$$

  - i – particle index
  - k – discrete time index
  - v – velocity of ith particle
  - x – position of ith particle
  - p – best position found by ith particle (personal best)
  - G – best position found by swarm (global best, best of personal bests)
  - γ1,2 – random numbers on the interval [0,1] applied to ith particle

## Theory - Equations ..

- The Common PSO Algorithm

$$v_i(k+1) = \phi(k)v_i(k) + \alpha_1\left[\gamma_{1i}\left(p_i - x_i(k)\right)\right] + \alpha_2\left[\gamma_{2i}\left(G - x_i(k)\right)\right]$$

  - ϕ - Inertia function
  - α1,2 – Acceleration constants

- As training progresses using a decreasing linear inertia function, the influence of past velocity becomes smaller.

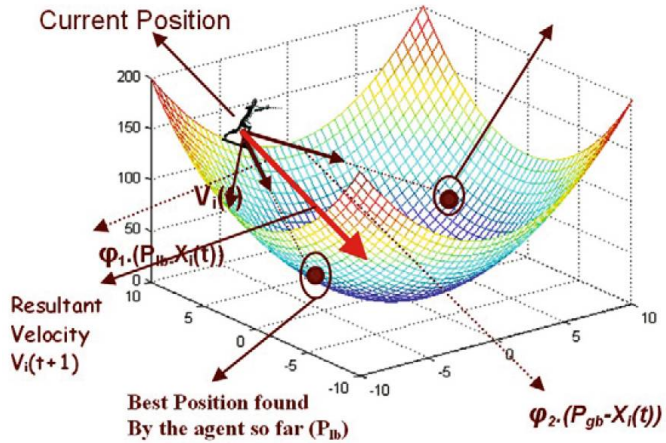



**Fig. 2.** Illustrating the dynamics of a particle in PSO

Pseudo Code

```
For each particle
{
    Initialize particle
}

Do until maximum iterations or minimum error criteria
{
    For each particle
    {
        Calculate Data fitness value
        If the fitness value is better than pBest
        {
            Set pBest = current fitness value
        }
        If pBest is better than gBest
        {
            Set gBest = pBest
        }
    }

    For each particle
    {
        Calculate particle Velocity
        Use gBest and Velocity to update particle Data
    }
```

## Code

- Matlab

```
clear all; close all; clc
%Function to be minimized
D=2;
objf=inline('4*x1^2-2.1*x1^4+(x1^6)/3+x1*x2-4*x2^2+4*x2^4','x1','x2');
objf=vectorize(objf);
%Initialization of PSO parameters
N=20; %population size (total function evaluations will be itmax*N)
itmax=30;
c1=1.05; c2=1.05;
wmax=1; wmin=0.3; %set to same value for constant w
w=linspace(wmax,wmin,itmax); %linear variation of w
%Problem and velocity bounds
a(1:N,1)=-1.9;  b(1:N,1)=1.9; %bounds on variable x1
a(1:N,2)=-1.1;  b(1:N,2)=1.1; %bounds on variable x2
d=(b-a);
m=a; n=b;
q=(n-m)/4; %initial velocities are 1/4 of parameter space size
%Random initialization of positions and velocities
x=a+d.*rand(N,D);
v=q.*rand(N,D);
%Evaluate objective for all particles
f=objf(x(:,1),x(:,2));
%Find gbest and pbest (in this case coincides with x)
[fgbest,igbest]=min(f);
gbest=x(igbest,:);
pbest=x; fpbest=f;
%Iterate
for it=1:itmax;
    %Update velocities and positions
    v(1:N,1:D)=w(it)*v(1:N,1:D)+c1*rand*(pbest(1:N,1:D)-x(1:N,1:D))+c2*rand*(repmat(gbest,N,1)-x(1:N,1:D));
    x(1:N,1:D)=x(1:N,1:D)+v(1:N,1:D);
    %Evaluate objectives
    f=objf(x(:,1),x(:,2));
    %Find gbest and pbest
    [minf,iminf]=min(f);
    if minf<= fgbest
        fgbest=minf; gbest=x(iminf,:);
    end
    inewpb=find(f<=fpbest);
    pbest(inewpb,:)=x(inewpb,:); fpbest(inewpb)=f(inewpb);
end %end loop on iterations
[gbest,fgbest]
```

Please note that this code actually works !

*Example 2.* Consider the two-dimensional function given by

$$f(\vec{x}) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2).$$

This function is known as the Rastrigin function [11] and has a global minimum value 0 at $x_1 = 0$ and $x_2 = 0$. A PSO is run to optimize the function. We used 30 particles and randomly initialized their positions and velocities in the interval $[-10, 10]$. We used the following parameter set-up: $C_1 = C_2 = 2.00$, $\omega = 0.729$, and the maximum particle velocity $V_{\max} = 10$. Figure 3 depicts the trajectory of the globally best particle towards the global minima of the function over different iterations.

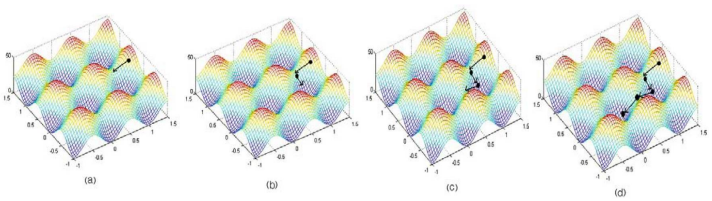
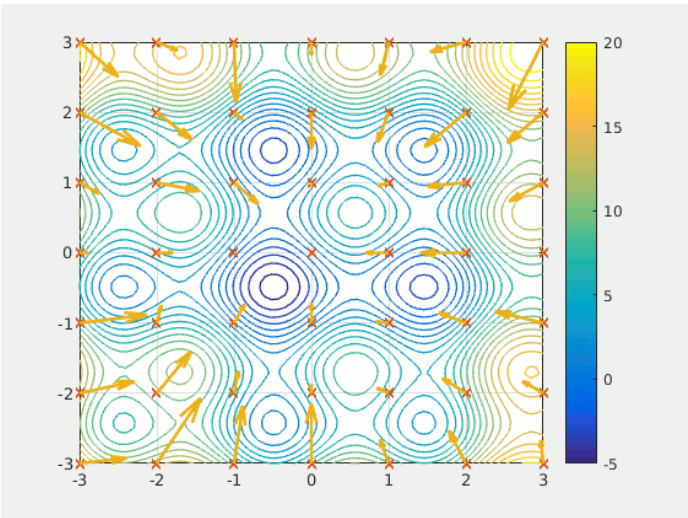
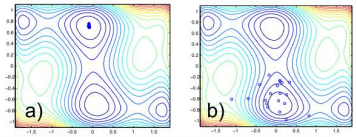



**Fig. 3.** Trajectory of the best particle for the 2-D Rastrigin function. (**a**) After 40 iterations (**b**) after 80 iterations (**c**) after 150 iterations (**d**) after 200 iterations



## What happens if…



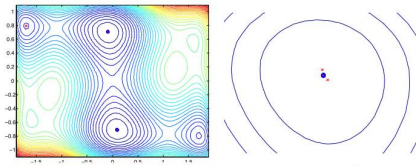

- Inertia (1000 runs, $|f_{opt}-f^*|$ )

| | w | N. | Iter. | Tot. | Max. | Avg. | Min. | Std. Dev. |
|---|---|---|---|---|---|---|---|---|
| 1) | 0,1 | 20 | 30 | 600 | 8,4 E-01 | 3,2 E-02 | 1,4 E-11 | 8,6 E-02 (b) |
| 2) | 0,5 | 20 | 30 | 600 | 8,2 E-02 | 2,4 E-03 | 1,5 E-12 | 4,5 E-02 |
| 3) | 1,0 | 20 | 30 | 600 | 2,4 E-01 | 5,0 E-03 | 9,8 E-07 | 1,2 E-02 |
| 4) | 2,0 | 20 | 30 | 600 | 1,0 E-00 | 1,6 E-01 | 1,1 E-04 | 1,6 E-01 |
| 5) | 1➔0,5 | 20 | 30 | 600 | 1.5 E-02 | 1,4 E-04 | 6,1 E-09 | 8,7 E-04 |
| 6) | 1➔0,3 | 20 | 30 | 600 | **6,9 E-04** | **4,7 E-06** | 4,6 E-12 | **3,3 E-05 (a)** |
| 7) | 1➔0,1 | 20 | 30 | 600 | 5,0 E-03 | 8,1 E-06 | 2,8 E-12 | 1,7 E-04 |

- $c_1 = c_2 = 1,05$
- $v_{ini} = \Delta x/4$

Case 6) has best worst case, best results on average, and is the most robust (lowest standard deviation) !

## What happens if…



- Population size (1000 runs, $|f_{opt}-f^*|$ )

| | N. | Iter. | Tot. | Max. | Avg. | Min. | Std. Dev. |
|---|---|---|---|---|---|---|---|
| 1) | 40 | 15 | 600 | 2,2 E-03 | 3,2 E-05 | 3,0 E-10 | **1,4 E-04** |
| 2) | 30 | 20 | 600 | 1,4 E-02 | 2,3 E-05 | 1,3 E-10 | 4,6 E-04 |
| 3) | 20 | 30 | 600 | 2,1 E-02 | **2,0 E-05** | 5,2 E-12 | 4,7 E-04 |
| 4) | 10 | 60 | 600 | 8,2 E-01 | 8,9 E-04 | 5,0 E-12 | 2,6 E-02 |
| 5) | 5 | 120 | 600 | 9,9 E-01 | 7,5 E-03 | 1,8 E-11 | 7,2 E-02 |

- $c_1 = c_2 = 1,05$
- w = lin. from 1,0 to 0,3
- $v_{ini} = \Delta x/4$

Small populations (red cross in figures) don't perform well.
Cases 1), 2) and 3) work well. Case 3) will be used in the following.

## What happens if…

- Cognitive component (1000 runs, $|f_{opt}-f^*|$ )

| $c_1$ | N. | Iter. | Tot. | Max. | Avg. | Min. | Std. Dev. |
|---|---|---|---|---|---|---|---|
| 0,0 | 20 | 30 | 600 | **6,9 E-03** | **1,9 E-05** | 1,5 E-12 | **4,4 E-04** |
| 0,5 | 20 | 30 | 600 | 1,9 E-02 | 2,5 E-05 | 6,2 E-13 | 6,1 E-04 |
| 1,05 | 20 | 30 | 600 | 2,5 E-01 | 2,6 E-04 | 2,1 E-11 | 8,0 E-03 |
| 1,5 | 20 | 30 | 600 | 8,6 E-01 | 8,3 E-04 | 8,6 E-13 | 2,6 E-02 |
| 2,0 | 20 | 30 | 600 | 6,2 E-02 | 1,6 E-03 | 1,4 E-11 | 3,9 E-02 |

- $c_2 = 1,05$
- w = lin. from 1,0 to 0,3
- $v_{ini} = \Delta x/4$

It seems that it would be better not to have the cognitive component at all (c1=0) but wait for next slide…

## What happens if…

- Social component (1000 runs, $|f_{opt}-f^*|$ )

| $c_2$ | N. | Iter. | Tot. | Max. | Avg. | Min. | Std. Dev. |
|---|---|---|---|---|---|---|---|
| 0,0 | 20 | 30 | 600 | 3,7 E-01 | 1,5 E-02 | 2,2 E-07 | 3,2 E-02 |
| 0,5 | 20 | 30 | 600 | 1,1 E-01 | 2,2 E-03 | 4,5 E-09 | 3,9 E-03 |
| 1,05 | 20 | 30 | 600 | 1,6 E-02 | 2,3 E-04 | 1,1 E-10 | 5,3 E-04 |
| 1,5 | 20 | 30 | 600 | **4,1 E-03** | **2,1 E-05** | 2,5 E-11 | **2,0 E-04** |
| 2,0 | 20 | 30 | 600 | 1,6 E-01 | 4,0 E-04 | 3,4 E-11 | 5,5 E-03 |

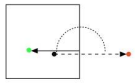
- $c_1 = 1,05$
- w = lin. from 1,0 to 0,3
- $v_{ini} = \Delta x/4$

The combined analysis of this and the preceeding table shows that $c_2$ should be bigger than $c_1$, but not too much…

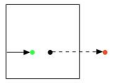
## The missing pieces

- What happens if a particle flies out of the parameter space?
  - Mirroring (a1)
  - Loop around (a2)
  - Random (a3)
- What happens if a particle violates constraints an becomes unfeasible?
  - Discard (b1)
  - Keep (b2)
- Integer/discrete variables
- Adaptivity and different random number distributions

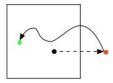
a1



a2



a3



b1


b2

## Conclusions

- PSO algorithm basics introduced
- Ease of Matlab implementation shown
- Numerical experiments:
  - Inertia should be linearly decreasing between 1 and 0,3
  - Population size matters (N between 20 and 30)
  - $c_2=c_1+\alpha$,with $\alpha$ between 1 and 1,5

# Algorithm Characteristics

- **Advantages**
  - Insensitive to scaling of design variables
  - Simple implementation
  - Easily parallelized for concurrent processing
  - Derivative free
  - Very few algorithm parameters
  - Very efficient global search algorithm
- **Disadvantages**
  - Tendency to a fast and premature convergence in mid optimum points
  - Slow convergence in refined search stage (weak local search ability)

**Improvements (I)**

### 3.3 The Constriction Factor $\chi$

In 2002, Clerc and Kennedy proposed an adaptive PSO model [17] that uses a new parameter '$\chi$' called the constriction factor. The model also excluded the inertia weight $\omega$ and the maximum velocity parameter $V_{\max}$. The velocity update scheme proposed by Clerc can be expressed for the $d$th dimension of $i$th particle as:

$$\left.\begin{aligned} V_{\mathrm{id}}(t+1) &= \chi[V_{\mathrm{id}}(t) + C_1 \cdot \varphi_1 \cdot (P_{\mathrm{id}}(t) - X_{\mathrm{id}}(t)) + C_2 \cdot \varphi_2 \cdot (g_{\mathrm{id}}(t) - X_{\mathrm{id}}(t))] \\ X_{\mathrm{id}}(t+1) &= X_{\mathrm{id}}(t) + V_{\mathrm{id}}(t+1), \end{aligned}\right\} \quad (2)$$

where,

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \quad \text{With} \quad \varphi = C_1 + C_2\,, \quad \varphi > 4.$$

Constriction coefficient results in the quick convergence of the particles over time. That is the amplitude of a particle's oscillations decreases as it focuses on the local and neighborhood previous best points. Though the particle converges to a point over time, the constriction coefficient also prevents collapse if the right social conditions are in place.

**Improvements (II)**

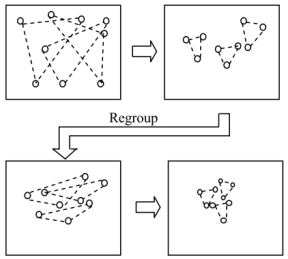
# Dynamic Multi-Swarm Particle Swarm Optimizer

**1) Small Sized Swarms**

Not as other evolutionary algorithms that prefer larger population, PSO needs a comparatively smaller population size. Especially for simple problems, a population with three to five particles can achieve satisfactory results. PSO with small neighborhoods performs better on complex problems.

**2) Randomly Regrouping Schedule**

Since the small sized swarms are searching using their own best historical information, they easily converge to a local optimum. If we keep the neighborhood structures unchanged, then there will be no information exchange among the swarms. In order to avoid this situation, every R generations, the population is regrouped randomly. In this way, the good information obtained by each swarm is exchanged among the swarms. This algorithm performs better on complex multimodal problems.



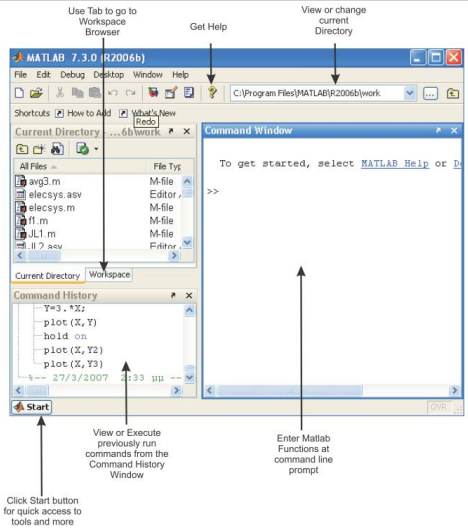

...

M2I BBS

# Modélisation et Simulation de Systèmes Biologiques

(#3 - intro Matlab, Freemat, Octave)

Georges Czaplicki, UPS / IPBS-CNRS
Tél. : 05.61.17.54.04, email : cgeorge@ipbs.fr




```
% IMPORTANT : la commande la plus importante : 'help'
```

```
% Pour voir les possibilités du Matlab : 'helpwin' ou 'demo'
```

**Introduction au langage Matlab** *(copie du "diary" ; "diary off" pour terminer)*

MATLAB always stores the result of the latest computation in a variable named *ans*. To store variables for a longer time we can specify our own names:

```
>> x = 5+2^2

x =

     9
```

and then we can use these variables in computations:

```
>> y = 2*x

y =

    18
```

These are examples of **assignment statements**: values are assigned to variables. **Each variable must be assigned a value before it may be used on the right of an assignment statement.** If you do not want to see the result of a statement in the Command Window, which is typically the case for intermediate statements, we can terminate the line with a semi–colon:

```
>> x = 5+2^2;
>> y = 2*x;
>> z = x^2+y^2

z =

   405
```

**Notes:**

- Other types of numerical variables can be defined explicitly if needed as:

  `char, single, int8, int16, int64, uint8, uint16, uint64.`

- MATLAB is accurate but does not do exact arithmetic !

```
>> 3*(5/3 - 4/3) - 1
ans =
     4.440892098500626e-16
```

- MATLAB identifies $i, j, pi$ also as numbers and to some extent $Inf$ and $NaN$:

```
>>  pi
ans = 3.1416

>> (-1)^(0.5)
ans = 0.0000 + 1.0000i

>> log(0)
ans =  -Inf

>> 0/0
ans = NaN
```

## Logical variables and operators

A logical variable can be "true" or "false", or one and zero in binary system.

```
>>S = 2>4
S =     0
```

Boolean algebra can be done in MATLAB using the logical operators AND (&) , OR (|) and NOT (~). For example given an arbitrary boolean variable $S$:

```
>> S | ~S
ans =     1
```

**Note**: MATLAB implicitly "casts" data types to avoid syntax errors:

- Logical values are converted to 1 and 0 when used as numbers.

```
>> (2 < 3) * 3
ans =     3
>> true * 3
ans =     3
```

- All numbers hold "true" boolean value when used in logical expressions, except for 0 itself which is "false".

```
>> false | -4.03
ans =    1
```

***Vecteurs :***

```
% vecteur en 3D
» v=[1 2 3]
v =
     1     2     3

% un autre vecteur
» w=[4 5 6]
w =
     4     5     6

% pas d'affichage
» w=[4 5 6];
```

```
% produit scalaire
» v*w
??? Error using ==> *
Inner matrix dimensions must agree.
» v*w'
ans =
    32
» w*v'
ans =
    32

% la norme
» norm(v)
ans =
    3.7417
» norm(w)
ans =
    8.7750
```

```
% vecteur en tant qu'une colonne
» z=[7;8;9]
z =
     7
     8
     9
» v*z
ans =
    50

% cos(phi) entre vecteurs v et z
» cosphi=v*z/(norm(v)*norm(z))
cosphi =
    0.9594

% angle en radians
» acos(cosphi)
ans =
    0.2859

% angle en degrés
» acos(cosphi)*180/pi
ans =
   16.3801
```

***Matrices***

```
% créer une matrice
» a=[1 2 3;4 5 6;7 8 9]
a =
     1     2     3
     4     5     6
     7     8     9

% plus facile de voir ce qu'on entre
» a=[1 2 3
4 5 6
7 8 9];

% un élément de la matrice
» a(1,3)

ans =
     3
```

```
% sous-matrice de a par extraction
» b=a(1,:)
b =
     1     2     3

» c=a(:,1)
c =
     1
     4
     7

% sous-matrice par suppression d'une colonne
» c=a;
» c(:,2)=[]

c =
     1     3
     4     6
     7     9
```

```
% certaines déclarations prédéfinies
» m=eye(3)

m =

     1     0     0
     0     1     0
     0     0     1

» m=ones(4,3)

m =

     1     1     1
     1     1     1
     1     1     1
     1     1     1

» m=zeros(3,4)

m =

     0     0     0     0
     0     0     0     0
     0     0     0     0
```

Concatenating matrices are straightforward MATLAB as long as their dimensions are consistent.

```
>> A = [1, 2, 3];B = [4, 5, 6];
>> C = [A, B]
C =
     1     2     3     4     5     6
>> C = [A; B]
C =
     1     2     3
     4     5     6
```

```
% opérations : produit matriciel
» x=a*a

x =
    30    36    42
    66    81    96
   102   126   150

% équivalent à :
» x=a^2

x =
    30    36    42
    66    81    96
   102   126   150
```

```
% produit élément par élément
» x=a.^2

x =
     1     4     9
    16    25    36
    49    64    81

% division élément par élément
» x=x./a

x =
     1     2     3
     4     5     6
     7     8     9
```

```
% matrice inverse
» y=inv(x)

Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 2.055969e-018.

y =

  1.0e+016 *

   -0.4504    0.9007   -0.4504
    0.9007   -1.8014    0.9007
   -0.4504    0.9007   -0.4504

% il y a un problème, mais essayons quand même de calculer
% le produit de cette matrice et de son inverse
» x*y

ans =

     2     0     0
     8     0    -4
    16     0     8

% le résultat ne donne pas la matrice d'identité. Pourquoi ?
```

```
» det(x)

ans =

     0

% C'est une matrice singulière !
```

```
% Quel est le rang de cette matrice ?
» rank(x)

ans =

     2
% les lignes (ou colonnes) sont linéairement dépendantes.
```

Comment vérifier la dépendance linéaire ?

$$\lambda_1 \cdot [1 \quad 2 \quad 3] + \lambda_2 \cdot [4 \quad 5 \quad 6] + \lambda_3 \cdot [7 \quad 8 \quad 9] = [0 \quad 0 \quad 0]$$

$$\begin{Bmatrix} 1\lambda_1 + 4\lambda_2 + 7\lambda_3 = 0 \\ 2\lambda_1 + 5\lambda_2 + 8\lambda_3 = 0 \\ 3\lambda_1 + 6\lambda_2 + 9\lambda_3 = 0 \end{Bmatrix} \rightarrow \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 \\ \lambda_{21} \\ \lambda_3 \end{bmatrix} = \mathbf{0}$$

$$[\lambda_1 \quad \lambda_2 \quad \lambda_{31}] = [1 \quad -2 \quad 1] \cdot const$$

```
% essayons encore une fois…
% changeons un seul élément : x(3,3)

» x=[1 2 3
     4 5 6
     7 8 10]

x =

     1     2     3
     4     5     6
     7     8    10

% le determinant…
» det(x)

ans =

    -3

% si det(x) n'est pas 0, alors le rang sera égal à 3
» rank(x)

ans =

     3
```

```
% l'inverse de la matrice x
» y=inv(x)

y =

   -0.6667   -1.3333    1.0000
   -0.6667    3.6667   -2.0000
    1.0000   -2.0000    1.0000

% voyons ce que ça donne…
» x*y

ans =

    1.0000         0   -0.0000
         0    1.0000         0
         0         0    1.0000

» y*x

ans =

    1.0000         0    0.0000
         0    1.0000         0
   -0.0000   -0.0000    1.0000
```

***Fonctions des matrices***

```
% Nous savons calculer l'expression f(x), où x est un chiffre
% (un scalaire). Mais comment faire si x est une matrice ?

% exemple : calculer exp(-A), où A est une matrice

» A=[4 2 1
     3 5 2
     2 4 6]

A =

     4     2     1
     3     5     2
     2     4     6
```

```
% décomposition de la matrice A en vecteurs et en valeurs propres

» [V,D]=eig(A)

  V =

      0.3284    0.4063    0.4560
      0.5430    0.3274   -0.7415
      0.7728   -0.8531    0.4922


  D =

      9.6605         0         0
           0    3.5121         0
           0         0    1.8273
```

```
% vérifions si la décomposition est correcte :
% est-ce que A est égale au produit V * D * inv(V) ?

» B=V*D*inv(V)

B =

    4.0000    2.0000    1.0000
    3.0000    5.0000    2.0000
    2.0000    4.0000    6.0000

% si oui, la différence devrait être 0

» A-B

ans =

  1.0e-014 *

   -0.5329    0.0444   -0.1332
    0.3553    0.0888    0.3553
    0.0222   -0.0888    0.0888
```

```
% si A=V*D*inv(V), alors f(A)=V*f(D)*inv(V)

% calculons exp(-Dii) sur la diagonale de D
» for i=1:3; D(i,i)=exp(-D(i,i));end;
» D

D =

    0.0001         0         0
         0    0.0298         0
         0         0    0.1608


% reconstruisons la matrice dans le repère initial
» C=V*D*inv(V)

C =

    0.0762   -0.0501    0.0029
   -0.0938    0.0884   -0.0222
    0.0430   -0.0630    0.0261

% Et maintenant la même chose, mais de façon plus simple
» expm(-A)

ans =

    0.0762   -0.0501    0.0029
   -0.0938    0.0884   -0.0222
    0.0430   -0.0630    0.0261
```

```
% Tableaux utiles : un vecteur ligne de 101 valeurs
% également espacées
» t=0:pi/100:pi;
» size(t)
ans =
     1   101


% à quoi correspond « pi » ?
» pi

ans =
    3.1416

% Attention : tous les nombres sont en double précision,
% seul le format de sortie varie

» format long

» pi

ans =
   3.14159265358979


» help format
```

```
FORMAT Set output format.
   All computations in MATLAB are done in double precision.
   FORMAT may be used to switch between different output
   display formats as follows:
     FORMAT         Default. Same as SHORT.
     FORMAT SHORT   Scaled fixed point format with 5 digits.
     FORMAT LONG    Scaled fixed point format with 15 digits.
     FORMAT SHORT E Floating point format with 5 digits.
     FORMAT LONG E  Floating point format with 15 digits.
     FORMAT SHORT G Best of fixed or floating point format with 5
                     digits.
     FORMAT LONG G  Best of fixed or floating point format with
                     15 digits.
     FORMAT HEX     Hexadecimal format.
     FORMAT +       The symbols +, - and blank are printed
                    for positive, negative and zero elements.
                    Imaginary parts are ignored.
     FORMAT BANK    Fixed format for dollars and cents.
     FORMAT RAT     Approximation by ratio of small integers.

   Spacing:
     FORMAT COMPACT Suppress extra line-feeds.
     FORMAT LOOSE   Puts the extra line-feeds back in.
```

```
% faisons le point sur les variables utilisées jusqu'à ici
» who
Your variables are:
a         b         cosphi    v         x
ans       c         t         w         z

» whos
  Name         Size         Bytes  Class
  a            3x3             72  double array
  ans          1x1              8  double array
  b            1x3             24  double array
  c            3x2             48  double array
  cosphi       1x1              8  double array
  t            1x101          808  double array
  v            1x3             24  double array
  w            1x3             24  double array
  x            3x3             72  double array
  z            3x1             24  double array
Grand total is 139 elements using 1112 bytes


% on peut tout nettoyer
» clear
» who
```

```
% graphiques 2D : comment tracer la courbe x=sin(t)
» t=0:pi/100:2*pi;
» x=sin(t);
» plot(t,x)

% toutes les informations sur 'plot' :
» help plot

PLOT   Linear plot.
PLOT(X,Y) plots vector Y versus vector X. If X or Y is a matrix,
then the vector is plotted versus the rows or columns of the
matrix, whichever line up.  If X is a scalar and Y is a vector,
length(Y) disconnected points are plotted.

PLOT(Y) plots the columns of Y versus their index.
If Y is complex, PLOT(Y) is equivalent to PLOT(real(Y),imag(Y)).
In all other uses of PLOT, the imaginary part is ignored.
```

```
Various line types, plot symbols and colors may be obtained with
PLOT(X,Y,S) where S is a character string made from one element
from any or all the following 3 colunms:

y     yellow       .     point              -     solid
m     magenta      o     circle             :     dotted
c     cyan         x     x-mark             -.    dashdot
r     red          +     plus               --    dashed
g     green        *     star
b     blue         s     square
w     white        d     diamond
k     black        v     triangle (down)
                   ^     triangle (up)
                   <     triangle (left)
                   >     triangle (right)
                   p     pentagram
                   h     hexagram

For example, PLOT(X,Y,'c+:') plots a cyan dotted line with a plus
at each data point; PLOT(X,Y,'bd') plots blue diamond at each data
point but does not draw any line.
```

```
% traçons la courbe en rouge et changeons les axes
» plot(t,x,'r')
» axis([0 2*pi -2 2])

% ajoutons une grille
» grid on
```

| Fonction | Description |
|---|---|
| ones(i,j) | crée un tableau de i lignes j colonnes contenant des 1 |
| zeros(i,j) | crée un tableau de i lignes j colonnes contenant des 0 |
| eye(i,j) | crée un tableau de i lignes j colonnes avec des 1 sur la diagonale principale et 0 ailleurs |
| toeplitz(u) | crée une matrice de Toeplitz symétrique dont la première ligne est le vecteur u |
| diag(u) | crée une matrice carrée avec le vecteur u sur la diagonale et 0 ailleurs |
| diag(U) | extrait la diagonale de la matrice U |
| triu(A) | renvoie la partie supérieure de A |
| tril(A) | renvoie la partie inférieure de A |
| linspace(a,b,n) | crée un vecteur de n composantes uniformément réparties de a à b |
| A\b | résolution du système linéaire `Ax=b` |
| cond(A) | conditionnement d'une matrice (norme euclidienne) |
| det(A) | déterminant d'une matrice |
| rank(A) | rang d'une matrice |
| inv(A) | inverse d'une matrice |
| pinv(A) | pseudo inverse d'une matrice |
| svd(A) | valeurs singulières d'une matrice |
| norm(A) | norme matricielle ou vectorielle |
| u' | prend le transposé de u |
| u*v | multiplication matricielle |
| u+v | addition matricielle |
| u-v | soustraction matricielle |
| u.* v | multiplication des tableaux u et v terme à terme |
| u./v | division du tableau u par le tableau v terme à terme |
| find(C(A)) | indices des composantes du tableau A vérifiant la condition C(A) |

TAB. 1 – Principales opérations sur les matrices.

## Making Plots

*Matlab* provides a variety of functions for displaying data as 2-D or 3-D graphics.

For 2-D graphics, the basic command is:

```
plot(x1, y1, 'line style', x2, y2, 'line style'...)
```

This command plots vector `x1` versus vector `y1`, vector `x2` versus vector `y2`, etc. on the same graph. Other commands for 2-D graphics are: **polar, bar, stairs, loglog, semilogx,** and **semilogy**.

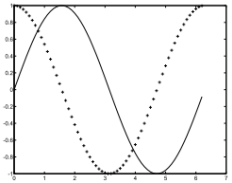
**Example 1:** Plot $y_1 = \sin(x)$ and $y_2 = \cos(x)$ with x in $[0, 2\pi]$ on the same graph. Use a solid line for $\sin(x)$ and the symbol + for $\cos(x)$. The first step is to define a set of values for `x` at which the functions will be defined.

```
x = 0 : 0.1 : 2*pi;
y1 = sin(x);
y2 = cos(x);
plot(x, y1, '-', x, y2, '+')
```

**Note**: Ordinarily *Matlab* prints the results of each calculation right away. Placing `;` at the end of each line directs *Matlab* to not print the values of each vector.

Another way to get multiple plots on the same graph is to use the **hold** command to keep the current graph, while adding new plots. Another **hold** command releases the previous one. For example, the following statements generate the same graph as in **Example 1**. *Matlab* remembers that the vector `x` is already defined.

```
plot(x, sin(x), '-')
hold
plot(x, cos(x), '+')
```

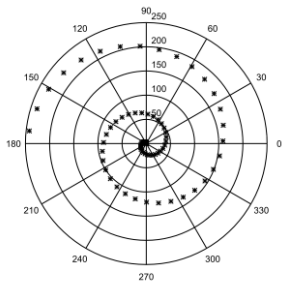


**Example 1**

The next example shows how *Matlab* generates a spiral using the polar coordinate system.

**Example 2:** Plot $\rho = \theta^2$ with $0 \leq \theta \leq 5\pi$ in polar coordinates.

```
theta = 0: 0.2: 5*pi;
rho = theta.^2;
polar(theta, rho, '*')
```



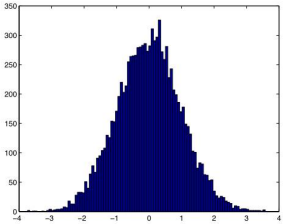

**Example 2**

## The hist command

Histogram plots are one of the other basic types of plots in MATLAB and can be produced using the hist function. The easiest form of using this function is hist(x), in which **x** denotes a vector. The command divides the range of the values in **x** to ten bins and plots the distribution of the element counts in each bin using bar plots.

An additional scalar parameter can be added afterwards in order to tune the number of bins, for example the following commands produce 10,000 normally distributed random numbers, plot their histogram using 100 bins:

```
>> x = randn(10000,1);
>> hist(x, 100)
```



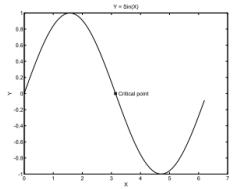
## Title and Labels

You can add a title and labels for the axes with the commands; **title, xlabel, ylabel** and **zlabel**. You can also add contour labels to a contour plot by the command **clabel**. Other text can be added to the graph by using the **text** or **gtext** commands. With **text**, you specify a location where left edge of a text string is placed. With **gtext**, you position the text string with the mouse.

Here is an example which adds titles and labels to the graph of $f(x) = sin(x)$.

**Example 4:** Plot $y = sin(x); 0 \leq x \leq 2\pi$, with appropriate labels.

```
x = 0: 0.1: 2*pi; plot(x,sin(x))
title('Y = Sin(X)')
xlabel('X');   ylabel('Y')
hold
plot(pi,0,'*')
text(pi + 0.1, 0, 'Critical point')  % or gtext('Critical point')
hold
```

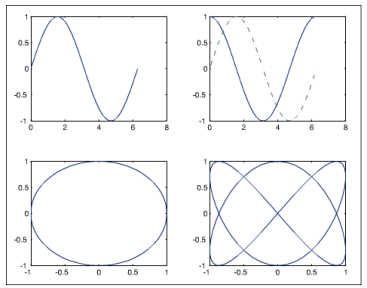



**Example 4**

- **Multiple Plots**

The command **subplot(m,n,p)** breaks the graph (or figure) window into an m-by-n matrix of small rectangular panes. The value of **p** is the pane for the next plot. Panes are numbered from left to right, top to bottom. To return to the default single graph per window, use either **subplot(1,1,1)** or **clf**.

You can have more than one graphics window on an X display. The *Matlab* command, **figure** opens a new window, numbering each new window. You can then use commands such as **clf, figure(h),** or **close** to manipulate the figure windows.

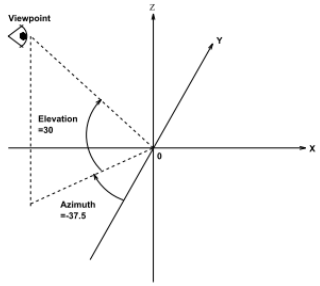


- **Viewpoint**

You can set the angle of view of a 3-D plot with the command:

```
view(az,el)
```

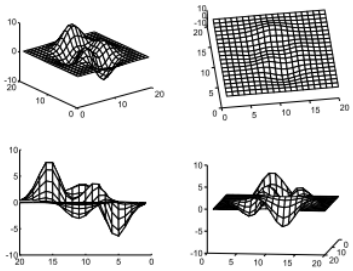
`az` is the azimuth and `el` is the elevation of the viewpoint, both in degrees. See the *viewpoint* figure for an illustration of azimuth and elevation relative to the Cartesian coordiate system.




**Default Viewpoint**

**Example 5:** View the internal *Matlab* **peaks** matrix from 4 different viewpoints. The first one, (**view(-37.5,30)**, is the default viewpoint.

```
subplot(2,2,1); mesh(peaks(20)); view(-37.5,30)
subplot(2,2,2); mesh(peaks(20)); view(-7,80)
subplot(2,2,3); mesh(peaks(20)); view(-90,0)
subplot(2,2,4); mesh(peaks(20)); view(-7,-10)
```



**Example 5**

- **Controlling Axes**

You can control the scaling and appearance of plot axis with the **axis** function. To set scaling for the x- and y- axes on the current 2-D plot, use this command:

```
axis([xmin xmax ymin ymax])
```

To scale the axes on 3-D plot, use this:

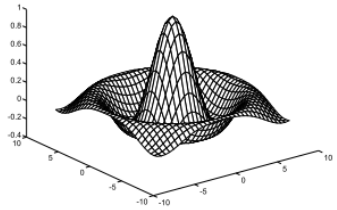
```
axis([xmin xmax ymin ymax zmin zmax])
```

In addition,

| | |
|---|---|
| **axis('auto')** | returns the axis scaling to its default where the best axis limits are computed automatically; |
| **axis('square')** | makes the current axis box square in size, otherwise a circle will look like an oval; |
| **axis('off')** | turns off the axes |
| **axis('on')** | turns on axis labeling and tic marks. |

```
% Graphiques 3D

» clear

% un exemple de la fonction sin(x)/x en deux dimensions (y=x).
% Tout d'abord, un vecteur ligne :
» x=-8:0.5:8;

% création de deux matrices carrées
% (meshgrid(x) = meshgrid(x,x)
» [X,Y]=meshgrid(x);

» whos
  Name      Size          Bytes  Class

  X        33x33           8712  double array
  Y        33x33           8712  double array
  x         1x33            264  double array
Grand total is 2211 elements using 17688 bytes

% MATLAB est sensible à la casse (x != X)
```

```
% calculons la distance par rapport au centre
% pour tous les nœuds de la grille
» R=sqrt(X.^2+Y.^2)+eps;

% que represente 'eps' ?
» eps
ans =
    2.220446049250313e-016

% calcul de la fonction pour le graphe
» Z=sin(R)./R;
» mesh(X,Y,Z)  (==> OK pour Matlab, mais pas pour FreeMat !)
» surf(X,Y,Z)
```



```
» help meshgrid

MESHGRID   X and Y arrays for 3-D plots.
[X,Y] = MESHGRID(x,y) transforms the domain specified by vectors
x and y into arrays X and Y that can be used for the evaluation
of functions of two variables and 3-D surface plots.
The rows of the output array X are copies of the vector x and
the columns of the output array Y are copies of the vector y.

[X,Y] = MESHGRID(x) is an abbreviation for [X,Y] = MESHGRID(x,x).
[X,Y,Z] = MESHGRID(x,y,z) produces 3-D arrays that can be used to
evaluate functions of three variables and 3-D volumetric plots.

For example, to evaluate the function  x*exp(-x^2-y^2) over the
range  -2 < x < 2,  -2 < y < 2,

     [X,Y] = meshgrid(-2:.2:2, -2:.2:2);
     Z = X .* exp(-X.^2 - Y.^2);
     mesh(Z)

MESHGRID is like NDGRID except that the order of the first two
input and output arguments are switched (i.e., [X,Y,Z] =
MESHGRID(x,y,z) produces the same result as [Y,X,Z] =
NDGRID(y,x,z)).  Because of this, MESHGRID is better suited to
problems in cartesian space, while NDGRID is better suited to N-D
problems that aren't spatially based.  MESHGRID is also limited to
2-D or 3-D.

See also SURF, SLICE, NDGRID.
```

```
% deux autres présentations possibles :
» surf(X,Y,Z)
» contour(X,Y,Z)

% impression ou sauvegarde dans un fichier
» print -dtiff -r300 Fig.tiff;

% ('help print' pour toutes les options)
```

For 3-D graphics, the most commonly used commands are:

```
plot3(x1, y1, z1, 'line style', x2, y2, z2, 'line style'...)
contour(x,y,Z)
mesh(x,y,Z), surf(x,y,Z)
```

The first statement is a three-dimensional analogue of `plot()` and plots lines and points in 3-D. The second statement produces contour plots of the matrix `Z` using vectors `x` and `y` to control the scaling on the `x-` and `y-` axes. For surface or mesh plots, you use the third statement where `x`, `y` are vectors or matrices and Z is a matrix. Other commands available for 3-D graphics are: **pcolor, image, contour3, fill3, cylinder,** and **sphere**.

| Instruction | Description |
|---|---|
| `plot(x,y)` | tracé de la courbe passant par les points `(x,y)` |
| `loglog(x,y)` | idem avec échelle logarithmique sur les deux axes |
| `semilogx(x,y)` | idem avec échelle logarithmique sur l'axe Ox |
| `semilogy(x,y)` | idem avec échelle logarithmique sur l'axe Oy |
| `plotyy(x,y,x,z)` | courbe (x,y) avec l'axe 0y à gauche, et courbe (x,z) avec l'axe 0z à droite |
| `xlabel('label')` | légende pour l'axe 0x |
| `ylabel('label')` | légende pour l'axe 0y |
| `title('label')` | titre au dessus du graphique |
| `legend('lab1','lab2','lab3',...)` | légende avec une chaîne de caractères pour chaque courbe |
| `text(x,y,'label')` | chaîne de caractères à la position x,y |
| `plot3(x,y,z)` | tracé de la surface passant par les points `(x,y,z)` |
| `hold on, hold off` | active/désactive la conservation de la fenêtre graphique à l'appel de la fonction plot |

TAB. 6 – Principales instructions graphiques.

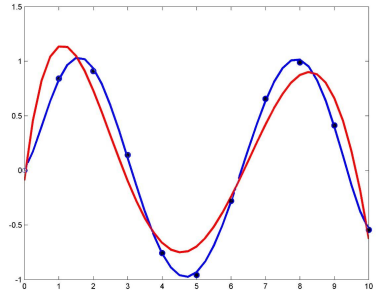
## Cas de données discrètes - interpolation

**Fonction `interp1`**

```
x  =  0:10;
y  =  sin(x);
xi =  0:.25:10;
yi =  interp1(x,y,xi);   (interplin1 pour FreeMat)
plot(x,y,'o',xi,yi)
```

**Fonctions `polyfit` et `polyval`**

```
x  =  0:10;
y  =  sin(x);
xi =  0:.25:10;
n  =  5;   (ordre du polynôme ; meilleur choix ?)
p  =  polyfit(x,y,n);
y5 =  polyval(p,xi);
n  =  7;
p  =  polyfit(x,y,n);
y7 =  polyval(p,xi);
plot(x,y,'o',xi,y5,'-r',xi,y7,'-b')
```




## MATLAB functions

Simple Built-in Functions

Here are some basic instances of the ready MATLAB functions:

- Trigonometric functions: `sin`, `cos`, `tan`, `asin`, `acos`, `atan` (their arguments should be in radians).
- Exponential: `exp`, `log`, `log2`, `log10`
- Random number generator: `rand`, `randn`, `randperm`, `mvnrnd`
- Data analysis: `min`, `max`, `mean`, `median`, `std`, `var`, `cov`
- Linear algebra: `norm`, `det`, `rank`, `inv`, `eig`, `svd`, `null`, `pinv`
- Strings: `strcmp`, `strcat`, `strfind`, `sprintf`, `sscanf`, `eval`
- Files: `save`, `load`, `csvwrite`, `csvread`, `fopen`, `fprintf`, `fscanf`
- Other: `abs`, `sign`, `sum`, `prod`, `sqrt`, `floor`, `ceil`, `round`, `sort`, `find`
- Master key: `help`, `doc`, and the external function Google!

***Scripts et fonctions***

```
% SCRIPT : programme sans arguments d'entrée ou de sortie,
%          utilise les données de l'espace de travail
% FONCTION : accepte des arguments et renvoie des valeurs,
%            les variables internes sont locales

% invocation de l'éditeur intégré
» edit

% taper dans la fenêtre de l'éditeur le contenu du programme :

% Programme traçant la surface z=x*y dans le carré [-1,1]x[-1,1]
x=-1:0.05:1;
y=x';
z=y*x;
mesh(x,y,z)
```

```
% sauvegarder sous le nom 'my_script.m'


% vérification
» type my_script

% Programme traçant la surface z=x*y dans le carré [-1,1]x[-1,1]
x=-1:0.05:1;
y=x';
z=y*x;
mesh(x,y,z)



% exécution du script
» clear
» my_script
» whos

  Name      Size         Bytes  Class
  x         1x41           328  double array
  y        41x1            328  double array
  z        41x41         13448  double array
Grand total is 1763 elements using 14104 bytes
```

```
% exemple de function : my_norme.m

» type my_norme

% Fonction calculant la norme du max d'un vecteur
function z=my_norme(x)
z=0;
for n=1:length(x)
   if(abs(x(n))) > z
      z=abs(x(n));
   end
end


% exécution
» clear
» x=[-1 3 -4 2];
» y=my_norme(x)
y =
     4

» whos
  Name      Size         Bytes  Class
  x         1x4             32  double array
  y         1x1              8  double array
Grand total is 5 elements using 40 bytes
```

## Fonctions en Matlab

### 1. M-files

```
function [y1, y2, …, ym] = toto(x1, x2, …, xn)
…
```

**Exemple avec une fonction utile (nargin) :**

```
function c  =  testarg1(a,b)
if  (nargin ==  1)
c  =  2*a;
elseif  (nargin  ==  2)
c  =  a  +  b;
end
```

| Instruction | Description |
|---|---|
| `nargin` | nombre d'arguments d'entrée d'une fonction |
| `nargout` | nombre d'arguments de sortie d'une fonction |
| `error` | interrompt l'exécution de la fonction, affiche le message d'erreur et retourne dans le programme appelant. |
| `warning` | imprime le message mais ne retourne pas dans le programme appelant |
| `pause` | interrompt l'exécution jusqu'à ce que l'utilisateur tape un `return` |
| `pause(n)` | interrompt l'exécution pendant $n$ secondes. |
| `pause off` | indique que les `pause` rencontrées ultérieurement doivent être ignorées, ce qui permet de faire tourner tous seuls des scripts requièrant normalement l'intervention de l'utilisateur. |
| `break` | sort d'une boucle `while` ou `for`. |
| `return` | retourne dans le programme appelant sans aller jusqu'à la fin de la fonction. |

TAB. 2 – Commandes de contrôle.

As a matter of convention, you should always call the m-file and the function by the same name. However, make sure that it does not conflict with a name that is already taken by another function. Otherwise the original function is "overwritten", i.e., cannot be called any more.

| Instruction | Description |
|---|---|
| `fzero(f,a)` | recherche des zéros d'une fonction `f` autour de `a` |
| `quad(f,a,b)` | calcul de l'intégrale d'une fonction `f` entre `a` et `b` |
| `spline(xx,yy)` | calcul de la spline cubique passant par les points (xx,yy) |
| `fft(a)` | transformation de Fourier rapide du vecteur `a` |
| `ode23(f,t,y0)` | résolution de l'équation `y'=f(t,x), y(0)=y0` |

TAB. 3 – Quelques algorithmes préprogrammés

A titre d'exemple nous détaillons ici la syntaxe pour utiliser la fonction `fzero`, pour trouver une racine d'une fonction d'une variable.
Syntaxe d'appel

```
x = fzero(fun,x0)
x = fzero(fun,x0,options)
x = fzero(fun,x0,options,P1,P2,...)
[x,fval] = fzero(...)
[x,fval,exitflag] = fzero(...)
[x,fval,exitflag,output] = fzero(...)
```

Description
- `fun` est une fonction inline ou 'fun' pour une fonction `Matlab` ou @fun pour un M_file.
- `x = fzero(fun,x0)` trouve un zéro près de `x0`, si `x0` est un scalaire. La valeur `x` renvoyée par `fzero` est près d'un point où fun change de signe ou bien NaN si la recherche a échouée.
- Si `x0` est un vecteur à 2 composantes, `fzero` le comprend comme un intervalle tel que `fun(x0(1))` et `fun(x0(2))` sont de signes opposés. Si ce n'est pas le cas il y a une erreur.
- `x = fzero(fun,x0,[],P1,P2,...)` permet de passer des paramètres supplémentaires P1, P2, etc,... à la fonction fun.

## 2. Fonctions inline

```
>>angle=inline('180*atan(y/x)/pi')
angle  =
Inline  function:
angle(x,y) =  atan(y/x)

>>angle(5,4)
ans   =
0.6747
```

## Flow control

Checking conditions: if, else, ifelse

Conditional statements check a given expression and based on the outcome execute certain parts of the code. You can also check several conditions in a sequential order with `elseif` statements.

```
if expression
    statements
elseif expression
    statements
elseif expression
    statements
...
else
    statements
end
```

Example: the following function computes the factorial $n! = \Pi_{k=1}^{n} k$ by recursive function calls.

```
function f = fac(n)
if n == 0
   f = 1;
else
   f = n*fac(n-1);
end
```

| français | test `Matlab` |
|---|---|
| et | & |
| ou | \| |
| non | ~ |
| égal | == |
| différent | ~= |
| plus petit que | < |
| plus grand que | > |
| plus petit ou égal à | <= |
| plus grand ou égal à | >= |

## Iterations: for and while loops

Iterations are defined by for and while loops. The difference between both is that for-loops have a fixed number of cycles. While-loops stop until a certain condition is matched. The for loop has the general syntax:

```
for variable = vector
    statements
end
```

Example: the following function computes the Fibonacci series.

```
function f = fibonacci(n)
f = zeros(n,1);
f(1:2) = [1 1];
for l=3:n
   f(l) = f(l-2) + f(l-1);
end
```

The while loop format is

```
while condition
    statements
end
```

Example: the following function sorts the elements of a vector in ascending order.

```
function v = gsort(v)
pos = 2;
while pos <= length(v)
  if v(pos)>v(pos-1) % move to the next position
    pos = pos + 1;
  else % swap the values
      foo = v(pos-1);
      v(pos-1) = v(pos);
      v(pos) = foo;
      if pos > 2 % decrease position
          pos = pos - 1;
      end
  end
end
```

For and while loops can be interrupted with the `break` statement. Generally, `for` and `while` loops should be avoided in MATLAB as they are very slow. Instead you should try to use built-in MATLAB functions or vectorize the code with the dot-operator.

| | |
|---|---|
| `dlmread('NomDeFichier','delimiteur')` | lecture du fichier |
| `dlmwrite('NomDeFichier',M,'delimiteur')` | écriture de M dans le fichier |
| `textread('NomDeFichier','format')` | lecture du fichier |
| `fid=open('NomDeFichier')` | ouverture du fichier NomDeFichier |
| `[A,count]=fscanf(fid,'format')` | lecture du fichier ouvert par open |
| `fprintf(fid,'format',données)` | écriture des données avec un format |
| `close(fid)` | fermeture |
| `fprintf('format',données)` | écriture des données avec un format |
| `fprintf('format',données)` | écriture des données avec un format |

TAB. 5 – Les commandes d'entrée-sortie et les accès fichiers

```
fid=fopen('NomDeFichier')    % ouvre le fichier
A = fscanf(fid,format)
[A,count] = fscanf(fid,format,size)
```

Lit les données avec le format spécifié. Un `format` est une chaine de caractères spécifiant en lecture le type de données à lire :

```
'%d' pour un entier
'%f' pour un réel
'%c'pour un caractère.
```

# Optimization

Optimization plays a central role in parameter estimation. Whenever you want to find a parameter set that leads to an optimal fit of your model to some measurements, you will have to employ some kind of optimization procedure.

## Finding roots and minima

Assume you want to find values for the arguments of a function for which it is zero. These are called the roots of a function. The `fzero` function can be used to find the root of a continuous function of one variable. You can either specify your own function

```
x = fzero(@myfun,x0);
```

where myfun is an M-file function that you defined before hand, or you specify an inline function (in MATLAB this is called an anonymous function) such as:

```
x = fzero(@(x)sin(x*x),x0);
```

Beside an input function, `fzero` requires a starting value, from where the search for the root starts. For example, the nearest roots of $\sin(x^2)$ starting from $x_0 = 2$ and $x_0 = 3$ are:

```
>> x = fzero(@(x)sin(x*x),2)

x =

    1.7725

>> x = fzero(@(x)sin(x*x),3)

x =

    3.0700
```

Note, that `fzero` only finds a root if your function is *continuous* and *crosses* the x-axis at the root (i.e. changes sign).

The minimum of a continuous function of one variable is computed by the `fminbnd` function:

```
x = fminbnd(fun,x1,x2)
```

where `x1` and `x2` are the boundaries within which the minimum is attained and `fun` is a function handle to an M-file function or a build in MATLAB function. For example,

```
>> x = fminbnd(@cos,3,4)

x =

    3.1416
```

find a numerical approximation to $\pi$. Similarly the minimum of an multi-variable, scalar function is computed by the `fminsearch` function.

```
x = fminsearch(fun,x0)
```

As an example consider the function $f(x, y) = ax^2 + by^2$ which has a uniquely defined minimum at $(x, y) = (0, 0)$ for $a, b > 0$. To solve for the minimum we need to define the function in a separate M-file.

```
function f = myfun(x,a,b)
f = a*x(1)^2 + b*x(2)^2;
```

Now the call

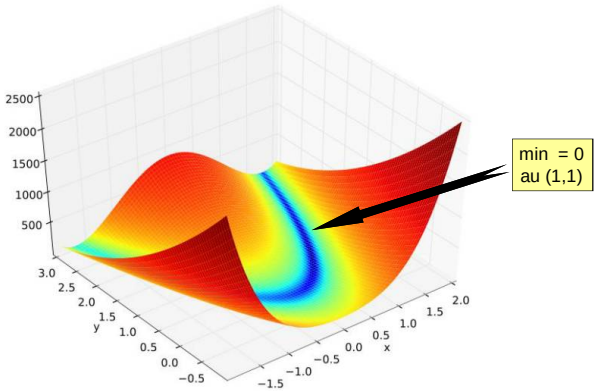
```
>> x = fminsearch(@(x) myfun(x,1,2),[0,1])

x =

   1.0e-03 *

    0.9298    0.0312
```

yields a numerical approximation to the true solution (starting at the value $(x, y) = (0, 1)$ and setting $a = 1, b = 2$). The result might come as a surprise, since we already started a the correct value of $x$ and end up wit a deviation in the order of $10^{-3}$. If we increase the default accuracy from $10^{-3}$ to $10^{-8}$ the deviation from the true value decreases accordingly.

```
>> opt = optimset('TolX',1e-8);
>> x = fminsearch(@(x) myfun(x,1,2),[0,1],opt)

x =

   1.0e-08 *

    0.1105    0.1613
```

The `optimset` function allows to adjust many more optimization settings. Consult the documentation for details!

## Exemple : minimisation de la fonction de Rosenbrock (banane)




$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2.$$

## Minimizing Rosenbrock's Function with fminsearch

```
banana = @(x)100*(x(2)-x(1)^2)^2+(1-x(1))^2;
```

```
[x,fval,exitflag] = fminsearch(banana,[-1.2, 1])
```

**This produces:**

```
x =
    1.0000 1.0000

fval =
    8.1777e-010

exitflag =
    1
```

**This indicates that the minimizer was found at [1 1] with a value near zero.**

Optimisation sous Matlab.

==========

Optimisation simple

==========

1) La fonction "fminsearch" ne marche pas. Même les exemples du manuel échouent :(

2) Minimisation d'une simple fonction :

- créer un fichier .m ou définir une fonction sur une ligne :

```
f = @(x,y) x.*exp(-x.^2-y.^2)+(x.^2+y.^2)/20;
ezsurfc(f,[-2,2])
```

- fonction passée à "fminunc" doit avoir un seul argument (qui peut être un vecteur) :

```
fun = @(x) f(x(1),x(2));
x0 = [-.5; 0];
```

- pour éviter le calcul des gradients, changer les options :

```
options = optimoptions('fminunc','Algorithm','quasi-newton');
(optionnellement : "options.Display = 'iter';")
```

- appeler la fonction fminunc :

```
[x, fval, exitflag, output] = fminunc(fun,x0,options);
```

- vérifier le retour, surtout output (info sur le bilan du calcul)

3) Minimisation dans le cas où il y a d'autres paramètres de la fonction :

- créer un fichier matlab :

```
function y = model(x,p)
% y= a * x^2 + b * x + c
y=p(1)*x*x+p(2)*x+p(3);
end
```

- appeler la fonction avec les paramètres après les options :

```
options = optimoptions('fminunc','Algorithm','quasi-newton','GradObj','off');
[x, fval, exitflag, output] = fminunc('model',[0],options,[1 -10 20]);
```

4) Exercice : fonction banane de Rosenbrock :

```
function y = banana(x)
y=100*(x(2)-x(1)^2)^2+(1-x(1))^2;

[x, fval, exitflag, output] = fminunc('banana',[-1.2 1],options);
```

- scruter les valeurs de sortie grad et hessian :

```
[x, fval, exitflag, output, grad, hessian] = fminunc('banana',[-1.2 1],options);
```

5) Comment passer les paramètres à la fonction :

- mettre les paramètres à la fin de la liste des arguments, et après les options dans l'appel :

```
function y = banana(x,a)
y=100*(x(2)-x(1)^2)^2+(a-x(1))^2;

[x, fval, exitflag, output,grad,hessian] = fminunc('banana',[-1.2 1],options,1);
```

- étudier les résultats de minimisation en fonction du paramètre "a".

## Ajustement à un modèle

6) Création de la fonction cible comme la somme des erreurs entre les valeurs calculées et mesurées :

- initialiser les données experimentales :

```
data=dlmread('data.dat')
```

| | |
|---|---|
| 1 | 11 |
| 2 | 4 |
| 3 | -1 |
| 4 | -4 |
| 5 | -5 |
| 6 | -4 |
| 7 | -1 |
| 8 | 4 |
| 9 | 11 |
| 10 | 20 |

- on va re-utiliser la fonction modèle qui décrit les donées (model.m)
- créer la fonction à minimiser :

```
function f = tgtfun(par,data)
n=size(data,1);
e=0;
for i=1:n
    x=data(i,1);
    ycal=model(x,par);
    yexp=data(i,2);
    dev=ycal-yexp;
    e=e+dev*dev;
end
f=e;
```

- lancer la minimisation :

```
[x,fval,exitflag,output,grad,hessian] = fminunc('tgtfun',[1,2,3],options,data)
```

- examiner la sortie (x = paramètres optimaux, i.e. a,b,c et non la coordonnée x !)

7) Introduction du bruit dans les données

- fonction "rand" : chiffre aléatoire générée uniformement entre (0,1)
- expression : "p*(2*rand-1)" varie entre (-p,p)
- si p est le pourcentage du bruit (e.g. 0.1 = 10%), alors on peut l'ajouter aux données :

```
p=0.1;
r=rand(size(data,1),1);
data(:,2)=data(:,2).*(1+p*(r(:,1).*2-1))
```

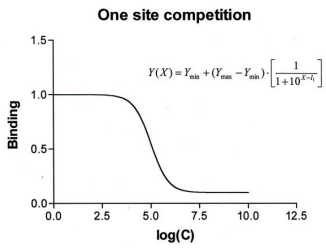
- refaire l'ajustement : quelle différence dans les valeurs finales des paramètres ?
- calculer l'écart type (sd=standard deviation) sur les paramètres optimisés :

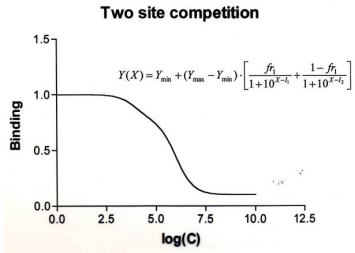
```
sd = sqrt(abs(diag(inv(hessian))))
```

- les erreurs au niveau de confiance de 95% sont données par 2*SD :

```
err = 2*sd
```

## Comparaison de deux modèles




1) fichier avec les données : compet.dat

```
data=dlmread('compet.dat')

0.00  98.02
0.71  103.99
1.43  96.65
2.14  96.77
2.86  97.22
3.57  87.00
4.29  61.55
5.00  40.00
5.71  20.58
6.43  8.11
7.14  15.33
7.86  11.88
8.57  6.77
9.29  12.41
10.00 9.97
```

$$Y(X) = Y_{min} + (Y_{max} - Y_{min}) \cdot \left[\frac{1}{1+10^{X-I_1}}\right]$$

2) premier modèle :

```
function y = compet1(x,par)
% one-site binding
a=par(1);
b=par(2);
ec50=par(3);
y = b + (a-b)/(1+10^(x-log10(ec50)));
```

$$Y(X) = Y_{min} + (Y_{max} - Y_{min}) \cdot \left[\frac{fr_1}{1+10^{X-I_1}} + \frac{1-fr_1}{1+10^{X-I_2}}\right]$$

3) deuxième modèle :

```
function y = compet2(x,par)
% two-site binding
a=par(1);
b=par(2);
ec50a=par(3);
f1=par(4);
ec50b=par(5);
y = b + (a-b)*f1/(1+10^(x-log10(ec50a))) + (a-b)*(1-f1)/(1+10^(x-log10(ec50b)));
```

NOTE: les paramètres devraient être de même ordre de grandeur, sinon problème de convergence.

4) fonction cible à minimiser :

```
function f = modcomp(par,data)
n=size(data,1);
e=0;
for i=1:n
   x=data(i,1);
   ycal=compet1(x,par);       % changer à compet2 pour le 2e modèle !
   yexp=data(i,2);
   dev=ycal-yexp;
   e=e+dev*dev;
end
f=e;
```

5) il faudra aussi la fonction ftest.m, disponible à l'adresse : opensiuc.lib.siu.edu/geol_comp/3/

6) Préparation des options de l'algorithme :

```
options = optimoptions('fminunc','Algorithm','quasi-newton','TolFun',1e-9);
```

7) Ajustement du premier modèle :

- mettre "compet1" dans la fonction modcomp

```
[x1, fval1] = fminunc('modcomp',[100,0,1e4],options,data)

x1 =
1.0e+04 *
0.0098    0.0012    3.3916

fval1 =
185.0000
```

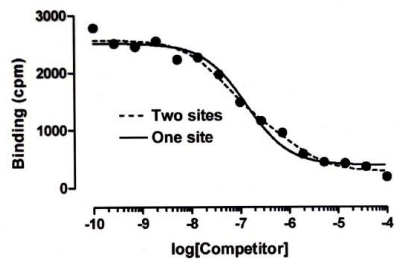
8) Ajustement au 2e modèle :

- mettre "compet2" dans la fonction modcomp

```
[x2, fval2] = fminunc('modcomp',[100,0,1e4,0.1,1e4],options,data)

x2 =
1.0e+05 *
0.0010    0.0001    0.1000    0.0000    1.0000

fval2 =
106.0824
```

9) Visualiser les résultats de deux ajustements

```
c=[0:0.1:10];
for k=1:length(c);v1(k)=compet1(c,x1);end;
for k=1:length(c);v2(k)=compet2(c,x2);end;
plot(data(:,1),data(:,2),'ok',c,v1,'-b',c,v2,'-r')
```



10) Faire le test de deux modèles :

```
[p,fv]=ftest(size(data,1),3,5,fval1,fval2)

p =

0.0819

fv =

3.3477
```

- interpretation : si p < 0.05, accepter le modèle #2

  si p > 0.05, accepter le modèle #1 (résultat du au hasard)

## Fitting functions with FreeMat

### FITFUN

Fits n (non-linear) functions of m variables using least squares and the Levenberg-Marquardt algorithm. The general syntax for its usage is

```
[xopt,yopt] = fitfun(fcn,xinit,y,weights,tol,params...)
```

Where **fcn** is the name of the function to be fit, **xinit** is the initial guess for the solution (required), **y** is the right hand side, i.e., the vector **y** such that:

$$xopt = \arg\min_x \|\mathrm{diag}(weights) * (f(x) - y)\|_2^2,$$

the output **yopt** is the function **fcn** evaluated at **xopt**. The vector **weights** must be the same size as **y**, and contains the relative weight to assign to an error in each output value. Generally, the ith weight should reflect your confidence in the ith measurement. The parameter **tol** is the tolerance used for convergence. The function **fcn** must return a vector of the same size as **y**, and **params** are passed to **fcn** after the argument **x**, i.e.,

$$y = fcn(x, param1, param2, ...).$$

Note that both **x** and **y** (and the output of the function) must all be real variables. Complex variables are not handled yet.

## Le code de la fonction banane de Rosenbrock :

```
function [f, J] = banana(x)
f = [100*(x(2)-x(1)^2)^2; (1-x(1))^2];
if nargout > 1, J=[-20*x(1) 10; -1 0]; end
```

## Conditions initiales et lancement :

```
x0 = [ -1.2 1 ];
opt = [ 1 1e-10 1e-12 100 1e-8 ];
[sol, info] = SMarquardt('banana', [], x0, tau);
```

## Résultats :

```
==> sol
ans =
    0.9995
    0.9990
==> info
ans =
    0.0000 0.0000 0.0001 0.0023 127.0000 1.0000 154.0000 13.0000
```

## Ordinary differential equations in MATLAB

Differential equations are the most common tool to model time and state-continuous dynamical processes. Many of these ODEs have to be analyzed numerically. MATLAB provides a powerful suite for the numerical integration of ODEs. We will focus on two types of ODE solvers: `ode45` and `ode15s`. `ode45` is the first method of choice, but whenever the ODE is stiff, i.e., the dynamics happen on very different time scales, you should use `ode15s` for integration. The general call of the two ODE solvers is

```
[t,x] = ode45(@rhs_handle,time_span,x0,options,args)
[t,x] = ode15s(@rhs_handle,time_span,x0,options,args)
```

where `rhs_handle` is a handle to the right hand side defining the ODE (see below). The vector defining the lower and upper integration boundaries is `time_span`, the vector `x0` defines the initial conditions. Options like relative and absolute error tolerances can be specified within the `options` structure, which is generated with the `odeset` function. However, they can also be omitted (with `[]`) to use standard options. All additional parameters (`args`) are passed to the `rhs_handle` file.

As a first example consider the logistic differential equation, which is a model for the growth of populations.

$$\dot{x}(t) = rx(t)(K - x(t))/K \tag{1}$$

The time-dependent population density is $x(t)$, parameter $r$ is the growth rate and $K$ is called the carrying capacity (Can you explain why?). In order to implement this equation in MATLAB we have to define the right-hand-side of equation 1 as a MATLAB function.

```
function dxdt = logistic(t,x,r,K)
dxdt = r*x*(K-x)/K;
```

We create this function using the MATLAB editor and save it in the current working directory under the file name `logistic.m`. Lets assume the following parameter values: $r = 2, K = 10$ and as initial conditions $x_0 = 0.1$. We can now integrate the ODE for the time interval $0 \leq t \leq 10$ and plot the results with the call

```
[t,x] = ode45(@logistic,[0 10],0.1,[],2,10);
```

Note that we have left the options structure empty as we want to use the standard integrator options. If we want to get the solution $x(t)$ for specific time points, e.g., at each 1/10 unit of $t$, we can set `tspan = [0:0.1:10]`. However, the internal step size of the integrator is chosen automatically and cannot be controlled with the `tspan` argument.

```
[t,x] = ode45(@logistic,[0 10],0.1,[],2,10);
plot(t,x)
```

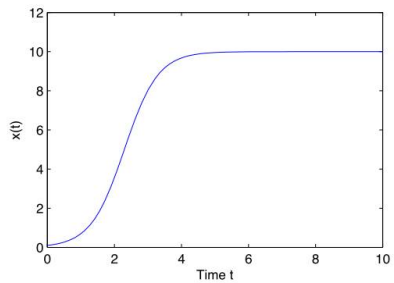
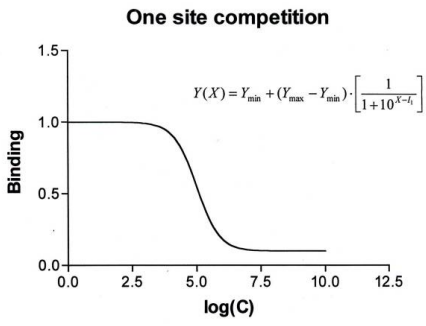
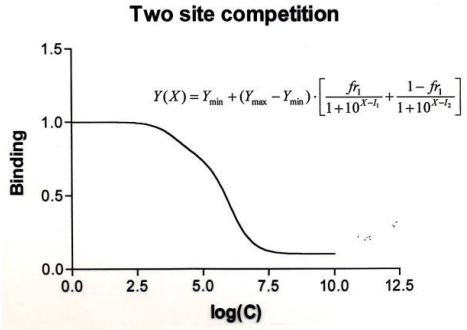



Figure 6: Solution of the logistic differential equation with $r = 2, K = 10$ and $x_0 = 0.1$.

## Octave : ajustement et comparaison de modèles




1) fichier avec les données : compet.dat

```
data=dlmread('compet.dat')

0.00  98.02
0.71  103.99
1.43  96.65
2.14  96.77
2.86  97.22
3.57  87.00
4.29  61.55
5.00  40.00
5.71  20.58
6.43  8.11
7.14  15.33
7.86  11.88
8.57  6.77
9.29  12.41
10.00 9.97
```

$$Y(X) = Y_{\min} + (Y_{\max} - Y_{\min}) \cdot \left[ \frac{1}{1+10^{X-I_1}} \right]$$

2) premier modèle :

```
function y = compet1(x,par)
% one-site binding
a=par(1);
b=par(2);
ec50=par(3);
y = b + (a-b)/(1+10^(x-log10(ec50)));
```

$$Y(X) = Y_{\min} + (Y_{\max} - Y_{\min}) \cdot \left[ \frac{fr_1}{1+10^{X-I_1}} + \frac{1-fr_1}{1+10^{X-I_2}} \right]$$

3) deuxième modèle :

```
function y = compet2(x,par)
% two-site binding
a=par(1);
b=par(2);
ec50a=par(3);
f1=par(4);
ec50b=par(5);
y = b + (a-b)*f1/(1+10^(x-log10(ec50a))) + (a-b)*(1-f1)/(1+10^(x-log10(ec50b)));
```

NOTE: les paramètres devraient être de même ordre de grandeur, sinon problème de convergence.

4) fonction cible à minimiser :

```
function f = modcomp(par,data)
n=size(data,1);
e=0;
for i=1:n
   x=data(i,1);
   ycal=compet1(x,par);       % changer à compet2 pour le 2e modèle !
   yexp=data(i,2);
   dev=ycal-yexp;
   e=e+dev*dev;
end
f=e;
```

5) il faudra aussi la fonction ftest.m, disponible à l'adresse : opensiuc.lib.siu.edu/geol_comp/3/

6) Optionnellement, préparation des options de la fonction fminunc :

```
options = optimset('TolX',1e-9,'TolFun',1e-9);
```

***options*** **is a structure specifying additional options. Currently,** ***fminunc*** **recognizes these options:**

**"FunValCheck",**
**"OutputFcn",**
**"TolX",**
**"TolFun",**
**"MaxIter",**
**"MaxFunEvals",**
**"GradObj",**
**"FinDiffType",**
**"TypicalX",**
**"AutoScaling".**

7) Ajustement au 1er modèle (mettre *compet1* dans *modcomp*)

```
x0=[100 0 1e4];

[x1,f1,inf1,out1,grad1,hess1]=fminunc(@(x) modcomp(x,data),x0,options)

x1 =
  9.8228e+001  1.1580e+001  3.3701e+004

f1 =  187.23

inf1 =  3

out1 =

  scalar structure containing the fields:
    iterations =  46
    successful =  40
    funcCount =  285

grad1 =

  -1.6655e-005  -3.3160e-006  9.3871e-009

hess1 =

  1.1226e+001  9.6423e-001  1.2908e-003
  9.6423e-001  1.6057e+001  1.5601e-003
  1.2908e-003  1.5601e-003  1.2037e-006
```

8) Ajustement au 2e modèle (mettre *compet2* dans *modcomp*)

```
X0=[100 0 1e4,0.1,1e4];

[x2,f2,inf2,out2,grad2,hess2]=fminunc(@(x) modcomp(x,data),x0,options)

x2 =
  9.9408e+001  1.0355e+001  1.1647e+004  5.8805e-001  1.6743e+005

f2 =  100.67

inf2 =  3

out2 =
  scalar structure containing the fields:
    iterations =  204
    successful =  188
    funcCount =  2083

grad2 =
  -2.0014e-004  1.4297e-004  -5.1629e-008  1.8568e-003  3.5202e-009

hess2 =
  1.3578e+001  1.8910e+000  3.4330e-003  -1.2879e+002  8.3938e-005
  1.8910e+000  1.6097e+001  1.7011e-003  -1.4745e+002  1.8178e-004
  3.4330e-003  1.7011e-003  3.9202e-006  -1.6398e-001  9.5930e-008
  -1.2879e+002  -1.4745e+002  -1.6398e-001  9.9174e+003  -8.4834e-003
  8.3938e-005  1.8178e-004  9.5930e-008  -8.4834e-003  1.0490e-008
```

## L'interprétation des paramètres de sortie

| | | |
|---|---|---|
| x1, x2 | = | valeurs optimales des paramètres de deux modèles |
| f1, f2 | = | valeurs résiduelles de la somme de déviations au carré |
| inf1, inf2 | = | codes de sortie (3 = critère ***TolFun*** satisfait) |
| out1, out2 | = | information détaillée sur le parcours de l'algorithme |
| grad1, grad2 | = | dernières valeurs de gradients |
| hess1, hess2 | = | Hessiens finaux |

## L'interprétation des paramètres de sortie (2)

```
norm(grad1)                    norm(grad2)
ans =   1.6982e-005            ans =  0.0018730
```

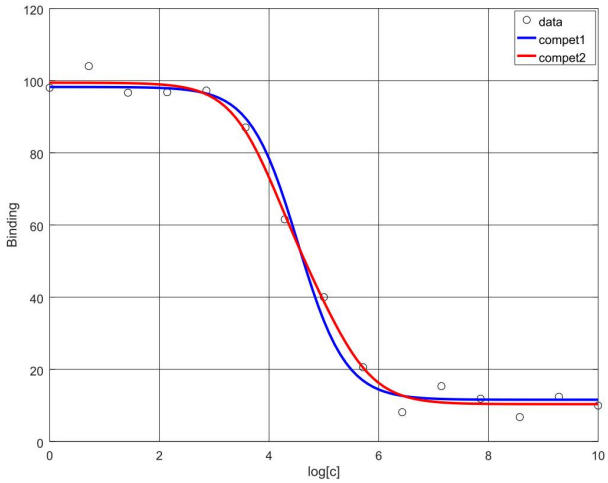
```
x1 = 9.8228e+001  1.1580e+001  3.3701e+004

x2 = 9.9408e+001  1.0355e+001  1.1647e+004  5.8805e-001  1.6743e+005
```

```
sigma1=abs(diag(inv(hess1)))

sigma1 =
  1.0198e-001
  7.1511e-002
  1.0825e+006

sigma2=abs(diag(inv(hess2)))
sigma2 =
  9.8059e-002
  7.7335e-002
  1.9066e+006
  1.7160e-003
  6.6328e+008
```

```
err1=1.96*sqrt(sigma1)

err1 =
  6.2590e-001
  5.2414e-001
  2.0392e+003

err2=1.96*sqrt(sigma2)

err2 =
  6.1376e-001
  5.4506e-001
  2.7064e+003
  8.1193e-002
  5.0478e+004
```

9) Visualisation des résultats du calcul

```
c=0:0.1:10;

for k=1:length(c);v1(k)=compet1(c(k),x1);end;
for k=1:length(c);v2(k)=compet2(c(k),x2);end;

plot(data(:,1),data(:,2),'ok',c,v1,'-b','linewidth',2,c,v2,'-r','linewidth',2)
```




10) Faire le F-test sur les résultats de deux modèles

```
[p,fv]=ftest(size(data,1),3,5,f1,f2)

p =  0.061289
fv =  3.8692
```

**Interprétation :**

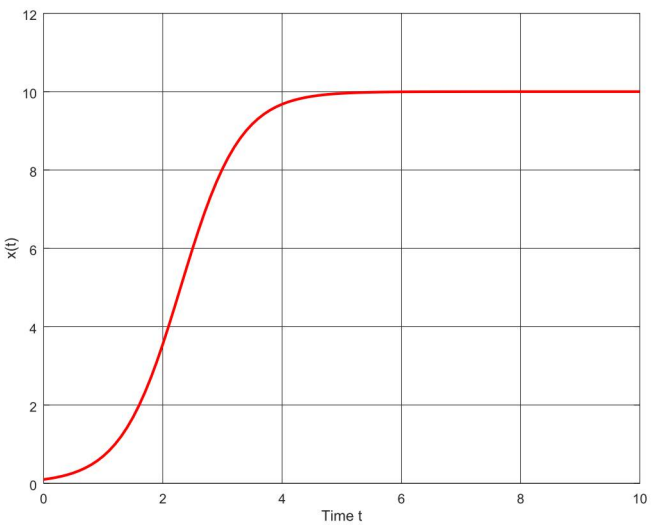
**Si p < 0.05, accepter le modèle 2**

**Si p > 0.05, accepter le modèle 1**

## Ordinary differential equations in Octave

```
type logistic
logistic is the user-defined function defined from:
E:\Octave\logistic.m

function dxdt=logistic(x,t,r,K)
  dxdt=r*x*(K-x)/K;
```

```
r=2;
K=10;
x0=0.1;
t=linspace(0,10,100);

x=lsode(@(x,t) logistic(x,t,r,K), x0, t);

plot(t,x,'-r',"linewidth",2)
grid on
xlabel('Time t')
ylabel('x(t)')
axis([0 10 0 12])
```



M2I BBS

# Modélisation et Simulation de Systèmes Biologiques

## (#4 - matrice de Leslie)

**Georges Czaplicki, UPS / IPBS-CNRS**
**Tél. : 05.61.17.54.04, email : cgeorge@ipbs.fr**

## Population Modeling Using the Leslie Matrix

This lab focuses on the use of the Leslie Matrix to determine the growth of a population, as well as the age distribution within the population over time. The model used here was described by P. H. Leslie in 1945.[1] This model has been used to describe the population dynamics of a wide variety of organisms including: brook trout, rabbits, lice, beetles, pine trees, buttercups, killer whales, and humans. We will apply the Leslie model to find the population and population distribution of a species of salmon that has been recently introduced into the area.

## Discussion

The Leslie model uses the following assumptions:

- We consider only the females in the salmon population.
- The maximum age attained by any individual salmon is three years.
- The salmon are grouped into three one-year age classes.
- An individual salmon's chances of surviving from one year to the next is a function of its age.
- The survival rate $P_i$ of each age group is known.
- The reproduction (fecundity) rate $F_i$ for each age group is known.
- The initial age distribution is known.

[1] *On the Use of Matices in Certain Population Mathematics*, Leslie, P.H., *Biometrika*, Volume XXXIII, November 1945, pp. 183–212

From these assumptions it is possible to construct a deterministic model by using matrices. Since the maximum age attained by any salmon is three years, the entire population can be broken up into three one-year age classes. Class 1 contains all salmon in their first year, class 2 contains all salmon in their second year, and class 3 contains all salmon in their third and last year of life.

Suppose we know the number of females in each of the three age classes at some time $t = t_0$. Let there be $x_1^{(0)}$ females in the first age class, $x_2^{(0)}$ females in the second age class, and $x_3^{(0)}$ females in the third age class. With these three numbers we form the column vector $\mathbf{x}^{(0)}$.

$$\mathbf{x}^{(0)} = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_3^{(0)} \end{pmatrix}$$

We call $\mathbf{x}^{(0)}$ the *initial age distribution vector* or the age distribution vector at time $t = t_0$.

As time progresses, the number of females in each of the three age classes changes because of three biological processes: birth, death, and aging. By describing these three processes quantitatively, we shall see how to project the initial age distribution vector into the future.

We will observe the population at discrete one year time intervals defined as $t_0, t_1, t_2, t_3, \ldots$. The birth and death processes between two successive observation times may be described by the means of defining parameters called the *average reproduction rate* and the *net survival rate*.

Let $F_1$ be the average number of females born to a single female in the first age class, $F_2$ is the average number of females born to a single female in the second age class, and $F_3$ is the average number of females born to a single female in the third age class. Each $F_i$ is average reproduction rate of a single female in the $i$th age group.

Let $P_1$ be the fraction of females in the first age class that survive the year to live on into the second age class. Let $P_2$ be the fraction of females in the second age class that survive the year to live on into the third age class. There is no $P_3$. After the third year, all the salmon die after spawning, so none survive to live on into a fourth age class. In general,

$F_i$ is the average reproduction rate of a female in the $i$th age class,
$P_i$ is the survival rate of females in the $i$th age class.

By their definitions, $F_i \geq 0$ since the number of offspring produced cannot be negative. In the case of this salmon population, $F_1 = 0$ and $F_2 = 0$ because the salmon only produce offspring in their last year of life. Thus, only $F_3$ has a positive value. Also, $0 < P_i \leq 1$ for $i = 1, 2$, since we assume that some of the salmon must survive into the next age class. This is true except for the last age class, when all the salmon die after spawning.

We next define the age distribution $\mathbf{x}^{(k)}$ at time $t_k$ by

$$\mathbf{x}^{(k)} = \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \\ x_3^{(k)} \end{pmatrix},$$

where $x_i^{(k)}$ is the number of female salmon in the $i$-th age class at time $t_k$. Now, at time $t_k$, the number of salmon in the first age class $x_1^{(k)}$, are just those salmon born between time $t_{k-1}$ and $t_k$. The number of offspring produced by each age class can be calculated by multiplying the reproductive rate for the age class times the number of females in the age class. The sum of all these values gives the total number of offspring produced. Thus, we can write

$$x_1^{(k)} = F_1 x_1^{(k-1)} + F_2 x_2^{(k-1)} + F_3 x_3^{(k-1)} \tag{1}$$

which says that the number of females in age class 1 equals the number of daughters born to females in age class 1 between times $t_k$ and $t_{k-1}$, plus the number of daughters born to females in age class 2 between times $t_k$ and $t_{k-1}$, plus the number of daughters born to females in age class 3 between times $t_k$ and $t_{k-1}$. In this example, since salmon only produce offspring in their last year of life, $F_1 = 0$ and $F_2 = 0$, so we get the equation

$$x_1^{(k)} = 0x_1^{(k-1)} + 0x_2^{(k-1)} + F_3 x_3^{(k-1)}. \tag{2}$$

The number of females in the second age class at time $t_k$ are those females in the first age class at time $t_{k-1}$ who are still alive at time $t_k$, or mathematically $x_2^{(k)} = P_1 x_1^{(k-1)}$, the number of females in the third age class at time $t_k$ are those females in the second age class at time $t_{k-1}$ who are still alive at time $t_k$, or mathematically $x_3^{(k)} = P_2 x_2^{(k-1)}$. We end up with the following system of linear equations.

$$\begin{aligned} x_1^{(k)} &= F_1 x_1^{(k-1)} + F_2 x_2^{(k-1)} + F_3 x_3^{(k-1)} \\ x_2^{(k)} &= P_1 x_1^{(k-1)} \\ x_3^{(k)} &= P_2 x_2^{(k-1)} \end{aligned} \tag{3}$$

We can use matrices to rewrite this system of equations as

$$\begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \\ x_3^{(k)} \end{pmatrix} = \begin{pmatrix} F_1 & F_2 & F_3 \\ P_1 & 0 & 0 \\ 0 & P_2 & 0 \end{pmatrix} \begin{pmatrix} x_1^{(k-1)} \\ x_2^{(k-1)} \\ x_3^{(k-1)} \end{pmatrix},$$

and even more compactly as

$$\mathbf{x}^{(k)} = L\mathbf{x}^{(k-1)}, \tag{4}$$

where

$$\mathbf{x}^{(k)} = \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \\ x_3^{(k)} \end{pmatrix}$$

is the age distribution vector at time $t_k$, and

$$\mathbf{x}^{(k-1)} = \begin{pmatrix} x_1^{(k-1)} \\ x_2^{(k-1)} \\ x_3^{(k-1)} \end{pmatrix}$$

is the age distribution vector at time $t_{k-1}$, and

$$L = \begin{pmatrix} F_1 & F_2 & F_3 \\ P_1 & 0 & 0 \\ 0 & P_2 & 0 \end{pmatrix} \tag{5}$$

is called the *Leslie Matrix*.

Finally, because $F_1 = F_2 = 0$, we get

$$L = \begin{pmatrix} 0 & 0 & F_3 \\ P_1 & 0 & 0 \\ 0 & P_2 & 0 \end{pmatrix}. \tag{6}$$

We can now generate a sequence of matrix equations to find the age distribution vector at any time $t_k$.

$$\begin{aligned} \mathbf{x}^{(1)} &= L\mathbf{x}^{(0)} \\ \mathbf{x}^{(2)} &= L\mathbf{x}^{(1)} = L(L\mathbf{x}^{(0)}) = L^2\mathbf{x}^{(0)} \\ \mathbf{x}^{(3)} &= L\mathbf{x}^{(2)} = L(L^2\mathbf{x}^{(0)}) = L^3\mathbf{x}^{(0)} \\ &\vdots \\ \mathbf{x}^{(k)} &= L\mathbf{x}^{(k-1)} = L(L^{k-1}\mathbf{x}^{(0)}) = L^k\mathbf{x}^{(0)} \end{aligned} \tag{7}$$

Thus, if we know the initial age distribution vector

$$\mathbf{x}^{(0)} = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_3^{(0)} \end{pmatrix}$$

and the Leslie matrix $L$, we can determine the female age distribution vector at any later time by multiplying an appropriate power of the Leslie matrix with the initial age distribution vector $\mathbf{x}^{(0)}$.

## An Example Using MATLAB

Suppose there are 1,000 females in each of the three age classes, so

$$\mathbf{x}^{(0)} = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_3^{(0)} \end{pmatrix} = \begin{pmatrix} 1,000 \\ 1,000 \\ 1,000 \end{pmatrix}.$$

Suppose further that the survival rate for salmon in the first age class is 0.5%, the survival rate for salmon in the second age class is 10%, and that each female in the third age class produces 2,000 female offspring. Then $P_2 = 0.005$, $P_3 = 0.10$, and $F_3 = 2,000$. The corresponding Leslie matrix for this system is

$$L = \begin{pmatrix} 0 & 0 & 2000 \\ .005 & 0 & 0 \\ 0 & .10 & 0 \end{pmatrix}.$$

To find the age distribution vector after one year, we use the equation $\mathbf{x}^{(1)} = L\mathbf{x}^{(0)}$. We can use MATLAB to find $\mathbf{x}^{(1)}$. First, enter the initial age distribution vector and the Leslie matrix.

```
>> x0=[1000;1000;1000];
>> L=[0 0 2000;0.005 0 0;0 .10 0]
L =
  1.0e+003 *
         0         0    2.0000
    0.0000         0         0
         0    0.0001         0
```

Note that Matlab is using scientific notation. The `1.0e+003 *` prior to the display of the matrix indicates that you should multiply each entry of the resulting matrix by $1.0 \times 10^3$, effectively moving each decimal point three places to the right. Let's try a different format for our output (Type `help format` to get a complete list of Matlab's formatting possibilities).

```
>> format short g
>> L=[0 0 2000;0.005 0 0;0 0.10 0]
L =
         0         0      2000
     0.005         0         0
         0       0.1         0
```

The command `format short g` prompts Matlab to use the best of fixed or floating point format, making a decision on each entry of the matrix, rather than applying one format to the entire matrix. Now, compute $x^{(1)}$ in the following manner.

```
>> x1=L*x0
x1 =
   2000000
         5
       100
```

The age distribution vector $\mathbf{x}^{(1)}$ shows that after the first year there are 2,000,000 salmon in the first age class, 5 in the second age class, and 100 in the third age class. Use MATLAB to find the age distribution vector $\mathbf{x}^{(2)}$ after two years.

```
>> x2=L*x1
x2 =
     2e+005
      10000
        0.5
```

You can produce exactly the same result with

```
>> x2=L^2*x0
x2 =
     2e+005
      10000
        0.5
```

The age distribution vector $\mathbf{x}^{(2)}$ shows that after two years there are 200,000 salmon in the first age class, 10,000 in the second age class, and 0.5 in the third age class. In real life it is not possible to have 1/2 of a salmon. However, let's postpone this issue for a moment and continue with the computation of the population after three years.

```
>> x3=L*x2
x3 =
        1000
        1000
        1000
```

Again, rather than proceeding with consecutive iterations, one can proceed directly to the answer with

```
>> x3=L^3*x0
x3 =
        1000
        1000
        1000
```

Note that the salmon population has returned to its original configuration, with 1,000 fish in each age category. Use Matlab to perform at least four more iterations; i.e., find $x^{(4)}$, $x^{(5)}$, $x^{(6)}$, and $x^{(7)}$. What pattern do you see?

## The Graph of the Age Distribution Vector

One of the best ways to examine trends in population growth is to sketch the graph of the age distribution vector versus time. Also, it's often desirable to track a population for more than three or four years.

### Using For Loops

Iterating the equation $x^{(k)} = Lx^{(k-1)}$ in the manner above is inefficient. If you know ahead of time the precise number of times that you wish to perform the iteration, then using a `for` loop in Matlab is the most efficient method.

First, load your Leslie Matrix and the initial age distribution vector.

```
>> L=[0 0 2000;0.005 0 0;0 0.10 0];
>> x0=[1000;1000;1000];
```

Let's iterate the equation $x^{(k)} = Lx^{(k-1)}$ a total of 24 times which will produce 24 generations of the age distribution vector. The makers of Matlab recommend that you plan ahead and reserve space in the memory of your computer to store your results. Let's follow this recommendation and reserve space for the results of 24 iterations by creating a $3 \times 24$ matrix of zeros: three rows because each age distribution vector contains three rows, twenty four columns because we will generate 24 age distribution vectors.

```
>> X=zeros(3,24)
```

Next, place the initial age distribution vector in the first column of the matrix $X$.

```
>> X(:,1)=x0
```

Recall that the Matlab index notation, `X(:,1)`, is read "every row, first column." Consequently, the command `X(:,1)=x0` places the initial conditions, contained in `x0`, into the first column of the matrix `X`.

Calculate the second through twenty-fourth columns of the matrix $X$ by iterating the equation $x^{(k)} = Lx^{(k-1)}$ for $k$-values ranging from two through twenty four.

```
>> for k=2:24, X(:,k)=L*X(:,k-1); end
```

When the number of iterations needed are known in advance, Matlab's `for` loop is the ideal construct. Recall that `2:24` produces a row vector, starting at 2 and proceeding in increments of 1 until it reaches the number 24. Therefore, the command `for k=2:24` begins the loop with a $k$-value equal to 2. The next time through the loop a $k$-value of 3 is used. Iteration continues and the last time through the loop, a $k$-value of 24 is used. Note that the `end` command signals the end of the loop.

The command `X(:,k)=L*X(:,k-1)` warrants explanation. Recall that `X(:,k)` is read "matrix `X`, every row, $k$th column." Similarly, the command `X(:,k-1)` is read "matrix `X`, every row, $k-1$st column." Consequently, the command `X(:,k)=L*X(:,k-1)` forms the product of the Leslie matrix $L$ and the $k-1$st column of matrix `X` and stores the result in the $k$th column of matrix `X`, precisely the iteration we need (recall that $\mathbf{x}^{(k)} = L\mathbf{x}^{(k-1)}$).[2]

Once the iteration is complete, you can display the contents of the matrix `X` by entering `X` at the Matlab prompt and pressing the Enter key.

```
>> X =
  Columns 1 through 6
        1000       2e+006       2e+005        1000       2e+006       2e+005
        1000            5        10000        1000            5        10000
        1000          100          0.5        1000          100          0.5
  Columns 7 through 12
        1000       2e+006       2e+005        1000       2e+006       2e+005
        1000            5        10000        1000            5        10000
        1000          100          0.5        1000          100          0.5
  Columns 13 through 18
        1000       2e+006       2e+005        1000       2e+006       2e+005
        1000            5        10000        1000            5        10000
        1000          100          0.5        1000          100          0.5
  Columns 19 through 24
        1000       2e+006       2e+005        1000       2e+006       2e+005
        1000            5        10000        1000            5        10000
        1000          100          0.5        1000          100          0.5
```

Type `help plot` at the Matlab prompt and read the resulting helpfile. Pay particular attention to the following lines.

```
>> help plot

 PLOT   Plot vectors or matrices.
   PLOT(Y) plots the columns of Y versus their index.
```

However, the first *row* of the matrix $X$ contains the numbers of female salmon in the first age class (juveniles), the second *row* contains the second age class (subadults), and the third *row* contains the number of female salmon in the third and final age class (adults). We want to plot the *rows* of $X$ versus the index but `plot(X)` plots the *columns* of $X$ versus the index.[3] The solution: plot the *transpose* of $X$.

The following command will produce an image similar to that in **Figure 1**.

```
>> plot(X')
```

If the command `plot(X')` is supposed to plot each of the three columns of the matrix `X'`, then where are the graphs of the remaining two columns in **Figure 1**? If you look closely you can see a little activity near the $x$-axis in **Figure 1**. Note that the upper limit on the $y$ -axis in Figure 1 is $2 \times 10^6$. When there is such a wide range in the data (values as small as 1/2 and as large as 2,000,000) you can get a better picture by plotting the natural logarithm of the salmon populations versus the time. The following command produces an image similar to that in **Figure 2**.

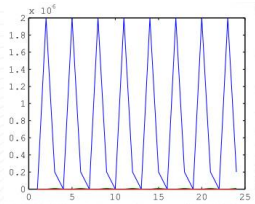



**Figure 1** The salmon population over time.

```
>> semilogy(X')
```

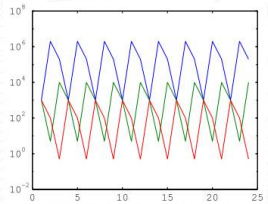



**Figure 2** Semilog plot of salmon population over time.

It is helpful to place a legend on your graph. The following command will produce an image similar to that in **Figure 3**.

```
>> legend('Juveniles','Subadults','Adults')
```

It is clear from the graph in **Figure 3** that each age division of the salmon population is oscillating with period 3.

**Extra for Experts.** The actual graph in **Figure 3** was produced using Matlab's handle graphics capabilities. If you are interested, try the following commands.

```
>> h=semilogy(X')
h =
        9.0011
        12.001
        13.001
>> set(h(1),'LineStyle','--')
>> set(h(2),'LineStyle',':')
>> legend('Juveniles','Subadults','Adults')
>> grid off
```

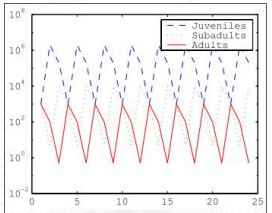



**Figure 3** Providing a legend.

In the latest release of Matlab, version 5.3, Release 11, you can change linestyles interactively by enabling plot editing, then right clicking a line with the mouse. A popup menu provides choices for linestyles, colors, and a host of other properties.

## Homework

*Instructions. For each of the following questions use the printer to produce a hardcopy image of the required graph.*

1. Suppose a particular species of salmon lives to *four* years of age. In addition, suppose that the survival rate of salmon in their first, second, and third years is 0.5%, 7%, and 15%, respectively. You also know that each female in the fourth age class produces 5,000 female offspring. The other age classes produce no offspring.

   a. Find the Leslie matrix for this population.

   b. If 1,000 female salmon in each of the four age classes are introduced into the system, find the initial age distribution vector.

   c. Use a `for` loop to iterate the Leslie equation 25 times. Use MATLAB to plot the natural logarithm of each age class of salmon versus time. What is the eventual fate of this salmon population?

   d. Calculate the salmon population on the 50th iteration, without calculating the preceding 49 iterations.

2. Suppose another species of salmon lives to *four* years of age. In addition, the survival rate of salmon in their first, second, and third years is 2%, 15%, and 25%, respectively. You also know that each female in the fourth age class produces 5,000 female offspring. The other age classes produce no offspring.

   a. Find the Leslie matrix for this population.

   b. If 1,000 female salmon in each of the four age classes are introduced into the system, find the initial age distribution vector.

   c. Use a `for` loop to iterate the Leslie equation 25 times. Use MATLAB to plot the natural logarithm of each age class of salmon versus time. What is the eventual fate of this salmon population?

   d. Calculate the salmon population on the 50th iteration, without calculating the preceding 49 iterations.

## Discussion

Although **equation 7** gives the age distribution of the population at any time, it does not immediately give a general picture of the dynamics of the growth process. To study the limiting behavior of the population growth we will need to learn about the eigenvalues and eigenvectors of the Leslie matrix. We will return to this problem later in the course.

## Properties of Leslie matrices

We list some properties of Leslie matrices (without proof), and we explore the effect of iterating the transition many times, that is, of allowing the population to pass through many reproductive cycles.

### Theorems about Leslie Matrices

1. A Leslie matrix **L** has a unique positive eigenvalue $\lambda_1$. This eigenvalue has multiplicity 1, and it has an eigenvector $\mathbf{x_1}$ whose entries are all positive.
2. If $\lambda_1$ is the unique positive eigenvalue of **L**, and $\lambda_i$ is any other eigenvalue (real or complex), then $|\lambda_i| \leq \lambda_1$. That is, $\lambda_1$ is a *dominant eigenvalue*.
3. If any two successive entries $\mathbf{a_j}$ and $\mathbf{a_{j+1}}$ of the first row of **L** are both positive, then $|\lambda_i| < \lambda_1$ for every other eigenvalue. That is, if the females in two successive age classes are fertile (almost always the case in any realistic population) then $\lambda_1$ is a *strictly dominant eigenvalue*.
4. Let $\mathbf{x^{(k)}}$ denote the state vector $\mathbf{L^k x^{(0)}}$ after **k** growth periods. If $\lambda_1$ is a strictly dominant eigenvalue, then for large values of **k**, $\mathbf{x^{(k+1)}}$ is approximately $\lambda_1 \mathbf{x^{(k)}}$, no matter what the starting state $\mathbf{x^{(0)}}$. That is, as **k** becomes large, successive state vectors become more and more like an eigenvector for $\lambda_1$.

**NOTE**: If all the entries of the eigenvector are negative, multiplication by -1 produces an eigenvector with all entries positive, without changing the validity of our reasoning.

Theorem 4 needs careful interpretation. It does not say that the sequence of states converges -- in particular, if the dominant eigenvalue is **> 1**, the sequence does not converge at all. On the other hand, if we "normalize" the state vector at each step -- say, by making its entries sum to 1 -- the sequence of modified state vectors does converge to an eigenvector. Normalized or not, the sequence shows us an *equilibrium age distribution* of the female population, which is approached over time.

## Case Study: Dynamical Systems and Spotted Owls

In this case study, we examine how eigenvalues and eigenvectors can be used to study the change in a population over time. We begin by recalling the example of the spotted owl given in the Introduction to Chapter 5.

The population of spotted owls is divided into three age classes: juvenile (up to 1 year old), subadult (1 to 2 years old), and adult (over 2 years old). The population is examined at yearly intervals. Since it is assumed that the number of male and female owls is equal, only female owls are counted in the analysis. If there are $j_k$ juvenile females, $s_k$ subadult females, and $a_k$ adult females at year $k$, then R. Lamberson et al. (see Reference 3) found that the population of owls could be modelled by the equation

$$\begin{bmatrix} j_{k+1} \\ s_{k+1} \\ a_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & .33 \\ .18 & 0 & 0 \\ 0 & .71 & .94 \end{bmatrix} \begin{bmatrix} j_k \\ s_k \\ a_k \end{bmatrix}$$

If we let $\mathbf{x}_k = (j_k, s_k, a_k)$, then we note that the population model has the form $\mathbf{x}_{k+1} = A\mathbf{x}_k$, which is a difference equation. This model is called the **stage-matrix model** for a population. The entries in the matrix $A$ have important meanings. The entries in the first row describe the **fecundity** of the population. Thus in the model above juveniles and subadults do not produce offspring, but each adult female produces (on the average) .33 juvenile females per year. The other entries in the matrix show **survival**. In this model, 18% of the juvenile females survive to become subadults, 71% of the subadults survive to become adults, and 94% of the adults survive each year. Note that the measures of fecundity and survival remain constant through time.

We wish to determine the long-term dynamics of the population: whether the population is becoming extinct or is increasing. To answer these questions we examine the eigenvalues of the matrix $A$ ($\lambda_1 = .98$, $\lambda_2 = -.02 + .21i$, $\lambda_3 = -.02 - .21i$). If we label the corresponding eigenvectors as $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$. we may express $\mathbf{x}_k$ as

$$\mathbf{x}_k = c_1(\lambda_1)^k\mathbf{v}_1 + c_2(\lambda_2)^k\mathbf{v}_2 + c_3(\lambda_3)^k\mathbf{v}_3$$

This expression of $\mathbf{x}_k$ is called the **eigenvector decomposition** of $\mathbf{x}_k$ (see page 337 of the text). Since each eigenvalue has magnitude less than 1, we conclude that $\mathbf{x}_k$ is approaching the zero vector as $k$ increases: the population is becoming extinct. Notice that the number of greatest importance to this analysis is $\lambda_1 = .98$, the eigenvalue of greatest magnitude. If $\lambda_1$ happened to be greater than 1, the population would instead be increasing steadily.

For example, consider Example 7 on page 345. If the survival rate for juveniles were somehow increased to 30%, the new matrix $A$ would be

$$\begin{bmatrix} 0 & 0 & .33 \\ .30 & 0 & 0 \\ 0 & .71 & .94 \end{bmatrix}$$

The eigenvalues of this matrix are $\lambda_1 = 1.01$, $\lambda_2 = -.03 + .26i$, $\lambda_3 = -.03 - .26i$. If we let $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ now denote eigenvectors of this new matrix, we will again have an eigenvector decomposition for $\mathbf{x}_k$:

$$\mathbf{x}_k = c_1(\lambda_1)^k\mathbf{v}_1 + c_2(\lambda_2)^k\mathbf{v}_2 + c_3(\lambda_3)^k\mathbf{v}_3$$

As $k \to \infty$ the second and third vectors tend to the zero vector, but the first does not. Thus $\mathbf{x}_k$ is approaching $c_1(1.01)^k\mathbf{v}_1$ as $k \to \infty$. So the population of owls would be increasing exponentially at a growth rate of 1.01; the population would be increasing by 1% per year. The eigenvector $\mathbf{v}_1$ gives the long-term distribution of the owls by life stages. In this case $\mathbf{v}_1$ is approximately $(10, 3, 31)$, so for every 31 adult females, there will be 10 juvenile females and 3 subadult females. We could further rescale $\mathbf{v}_1$ so that its entries sum to 1, namely: $(.227, .068, .705)$. The entries in this vector show the fraction of the owl population in each class; for example, 22.7% of the owls would be juveniles.

For convenience, here is a summary of our results drawn from Chapter 5.

1. We use the difference equation $\mathbf{x}_{k+1} = A\mathbf{x}_k$ to model the population in question; $A$ is an $n \times n$ matrix, where the population has been divided into $n$ classes or stages.
2. We find the eigenvalues of $A$ and list them in descending order of magnitude: $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$.
3. If $A$ is diagonalizable or if $A$ has distinct (possibly complex) eigenvalues, then we may express $\mathbf{x}_k$ in terms of its eigenvalues and corresponding eigenvectors:
$$\mathbf{x}_k = c_1(\lambda_1)^k\mathbf{v}_1 + c_2(\lambda_2)^k\mathbf{v}_2 + \ldots + c_3n(\lambda_n)^k\mathbf{v}_n$$
4. If $|\lambda_1| < 1$, then the population is decreasing to extinction.
5. If $\lambda_1$ is a real number greater than 1 and all the other eigenvalues are less than 1 in magnitude, then the population is increasing exponentially. As noted on page 338, in this case the eigenvector $\mathbf{v}_1$ gives the stable distribution of the population between classes, and yields the percenatges found in each class if scaled so that its entries sum to 1.

**Exercice :**

In the 1930's (before its virtual extinction and a great change in its survival rates) a researcher studied the blue whale population (see References 2, 5 and 6 for this data). Due to the long gestation period, mating habits, and migration of the blue whale, a female can produce a calf only once in a two-year period. Thus the age classes for the whale were assumed to be: less than 2 years, 2 or 3 years, 4 or 5 years, 6 or 7 years, 8 or 9 years, 10 or 11 years, and 12 or more years. The matrix for the model is given by

$$\begin{bmatrix} 0 & 0 & .19 & .44 & .50 & .50 & .45 \\ .77 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & .77 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .77 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & .77 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & .77 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & .77 & .78 \end{bmatrix}$$

Determine whether the blue whale population is becoming extinct in this model. If the population is not becoming extinct, determine the percentage of each class in the stable population.

**Project : Leslie Matrix or Population Projection Matrix**

The following table lists reproduction and survivor rates for the female population of a certain species of domestic sheep in New Zealand. (For animal populations, it is conventional to consider only the females, since only they reproduce, and they are usually a fixed percentage of the total population.) Sheep give birth only once a year, which dictates a natural time step of one year. In the species under consideration, sheep seldom if ever live longer than 12 years, which gives a natural stopping point for the age class.

**Birth and Survival Rates for Female New Zealand Sheep**

| age(years) | birth rate | survival rate |
|---|---|---|
| 0-1 | 0.000 | 0.845 |
| 1-2 | 0.045 | 0.975 |
| 2-3 | 0.391 | 0.965 |
| 3-4 | 0.472 | 0.950 |
| 4-5 | 0.484 | 0.926 |
| 5-6 | 0.546 | 0.895 |
| 6-7 | 0.543 | 0.850 |
| 7-8 | 0.502 | 0.786 |
| 8-9 | 0.468 | 0.691 |
| 9-10 | 0.459 | 0.561 |
| 10-11 | 0.433 | 0.370 |
| 11-12 | 0.421 | 0.000 |

**(2) Activities:**

(a) Enter the Leslie matrix *L* for the table given above.
(b) Compute the eigenvalues of *L*. How many real eigenvalues are there?
(c) Find the eigenvectors for the positive eigenvalues. Modify your eigenvectors (if necessary) to make it a state vector, and describe what it tells you about a distribution into age classes. (**Hint**: Any multiple of an eigenvector is also an eigenvector. Your *state* vector should have entries which sum to 1.)
(d) If the state vector you found in the previous step is the age distribution in a given year, what will the distribution be in one year? In ten years?

**Matrice de Leslie : la récolte**

## Harvesting an Age-Distributed Population

**Part 1: Sustainable harvesting**

Recall that in any particular year, a population (e.g., a single herd of New Zealand sheep) can be represented by a *state vector* $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{11}, \mathbf{x}_{12})^T$, where $\mathbf{x}_i$ represents the number of female animals in the **i**-*th* age class. If absolute numbers are not known, a state may be represented equally well by a vector of *fractions* of the population in each age class, i.e., by a vector whose entries sum to 1. The *Leslie growth matrix* for the population is the transition matrix **L** from the state in one year to the state in the next year. The entries $\mathbf{a}_1, \ldots, \mathbf{a}_{12}$ of the first row of **L** represent the rates of birth for each age class, while the subdiagonal entries $\mathbf{b}_1, \ldots, \mathbf{b}_{11}$ represent the survival rates for each class. Thus, if **x** is the state vector in a given year, the state vector after one year's growth is **Lx** and the *growth* in that year (distributed in age classes) is **Lx - x**.

In the Leslie Growth Models module, we saw that a New Zealand sheep population will increase by about 17.6% per year -- and approach a stable age distribution -- if left alone to do nothing but reproduce (and perhaps get sheared once in a while). However, New Zealand's sheep farmers cannot live entirely on their income from wool, especially if they have to keep feeding ever more sheep. A desirable goal for management of a sheep herd (or any renewable resource) is to find a stable configuration from which one can harvest the *growth* at regular intervals -- thereby producing income and returning the population to its previous configuration.

A *sustainable harvesting policy* is a plan for harvesting on a regular schedule in such a way that the harvest is always the same and the state of the population after harvesting is always the same.

Suppose we let $\mathbf{h_i}$ be the fraction of the **i**-th age group that will be harvested at the end of each growth period, and we let **H** be the diagonal matrix whose entries are the $\mathbf{h_i}$'s. If we start a growth period with age-distribution state **x**, then the state after growth will be **Lx**. The harvest after growth will be **HLx**, and that will reduce the population to **Lx - HLx**, or **(I - H)Lx**. To be sustainable, the population state after harvest must match the starting state, i.e., **(I - H)Lx = x**. That is, **x** must be an eigenvector for eigenvalue **1** for the matrix **(I - H)L**. In this part of the module, we explore some of the implications of this observation.

1. Enter the Leslie matrix **L** in symbolic form (i.e., without specific numbers assigned to the **a**'s and **b**'s), and compute **(I - H)L**. You should find that **(I - H)L** is another Leslie matrix. It differs from **L** in that the **i**-th row of **L** has been mulitplied by $\mathbf{1 - h_i}$.
2. Recall that the dominant eigenvalue $\lambda_1$ of a Leslie matrix is the unique positive eigenvalue of **L**. Now think about the eigenvalues of the Leslie matrix **(I - H)L**, in which the **i**-th row of **L** has been multiplied by $\mathbf{1 - h_i}$. The characteristic polynomial is a single equation in 12 unknowns (the **h**'s) that can be satisfied in many different ways. Thus, there are infinitely many ways to construct a sustainable harvesting policy. In what follows, we consider three of those ways.

3. Before we continue, we need to recompute the dominant eigenvalue $\lambda_1$ for the Leslie matrix associated to our New Zealand sheep population. Evaluate them and specify $\lambda_1$.
4. A *uniform* harvesting policy is one in which the same fraction **h** is harvested from each age group. In this case, we must have **(1 - h)Lx = x**. Explain why this means that **h** must satisfy $\lambda_1 = \mathbf{1/(1-h)}$, so $\mathbf{h = 1 - 1/}\lambda_1$. Use this observation to find a uniform fraction of the New Zealand sheep population that can be harvested every year and leave the population distribution the same at the start of each year.
5. Explain why the harvest rate is not the same as the growth rate of approximately 17.6%.

**Part 2: Harvesting the youngest class**

Sheep are not all equally valuable for harvest -- in fact, in world meat markets, lamb is much more valuable than mutton. Thus, the best economic use of the herd might be to harvest only lambs and keep the mature ewes alive to breed more lambs. Is there a sustainable harvesting policy for this case?

If we harvest only from the youngest group, then $\mathbf{h_1 = h}$, the fraction harvested from that group, and all the other **h**'s are zero. In step 2 of Part 1, we asked you to think about the algebraic condition on **h**'s in order to have **1** as an eigenvalue of **(I - H)L**. With only the first **h** different from zero, that condition simplifies to

$$(1 - h)(a_1 + a_2b_1 + a_3b_1b_2 + \ldots + a_{12}b_1b_2b_3\ldots b_{11}) = 1$$

or

$$(1 - h)R = 1,$$

where

$$R = a_1 + a_2b_1 + a_3b_1b_2 + \ldots + a_{12}b_1b_2b_3\ldots b_{11}.$$

The quantity **R** is called the *net reproduction rate* of the population -- it is, in fact, the average number of daughters born to a ewe in her expected lifetime.

1. Calculate the net reproduction rate **R** for the New Zealand sheep population.
2. To have a sustainable harvesting policy with only lambs being harvested, we must have **(1 - h)R = 1**. Find the number **h** that satisfies this condition. What fraction of the lambs should be harvested each year?
3. Find the stable population distribution **x** for this harvesting policy. What fraction of the total population should be lambs at the start of the growth period? At the end of the growth period? What fraction of the total population is harvested after the growth period? How does this compare with uniform harvesting?

**Part 3: Optimal harvesting**

We have now considered two sustainable harvesting policies: harvesting the same fraction from every age group, and harvesting only from the youngest age group. The latter probably produces a much smaller harvest in pounds of meat, but possibly a more valuable harvest for the sheep farmer. We now ask: What sustainable harvesting policy would produce the largest possible harvest in terms of numbers of animals? Another way to ask the question: What combination of harvesting fractions $\mathbf{h_1, h_2, ..., h_{12}}$ results in the largest fraction of the total population being harvested -- while still maintaining sustainability?

On the face of it, this is a very difficult question. However, there is a theorem from linear programming theory that simplifies the question considerably:

**Optimal Sustainable Yield**

If a sustainable harvesting policy is optimal, it harvests only from one or two age classes. If two age classes are harvested, then the older class is completely harvested.

[C. Rorres, ``Optimal Sustainable Yield of a Renewable Resource," *Biometrics*, Vol. 32, 1976, pages 945-948.]

1. For the New Zealand sheep population, it can be shown (by techniques beyond the level of this course) that the optimal yield is achieved when $\mathbf{h_1 = 0.522}$, $\mathbf{h_9 = 1}$, and all the other h's are zero. Find the sustainable age distribution for this policy, both before and after growth.
2. What fraction of the total population is harvested each year? How does this compare with uniform harvesting and with lambs-only harvesting?

**Points to ponder:**

1. Describe in your own words the meaning of "sustainable harvesting policy." How does a second Leslie matrix for a given population arise in this context? What is the significance of 1 as an eigenvalue for that matrix? What is the significance of an eigenvector for the eigenvalue 1?
2. If a sustainable harvesting policy takes the same fraction from every age group, how is that fraction calculated from the original Leslie matrix? How is a stable age distribution calculated?
3. If a sustainable harvesting policy takes only a fraction of the youngest age group, how is that fraction calculated from the original Leslie matrix? How is a stable age distribution calculated?
4. Can either uniform harvesting or youngest-only harvesting be an optimal sustainable policy in the sense of harvesting the largest number of individuals? Explain.

M2I BBS

# Modélisation et Simulation de Systèmes Biologiques

## (#5 – équa log & chem)

**Georges Czaplicki, UPS / IPBS-CNRS**
**Tél. : 05.61.17.54.04, email : cgeorge@ipbs.fr**

## Equations logistiques

Suppose living cells grow in a nutrient-rich medium. If ***X*** is the concentration of the biomass, then ***dX/dt*** is the ***growth rate***, or change in biomass per unit time.

Usually the change of cellular mass is directly related to the amount of digested nutrients. This, in turn, is proportional to the biomass concentration. The growth rate is then given by the Malthus law (1798):

$$\frac{dX}{dt} = \mu X$$

where the constant of proportionality $\mu$ is the ***per capita* growth rate**:

$$\mu = \frac{1}{X}\frac{dX}{dt}$$

The solution of the above equation is an exponential curve:

$$X(t) = X_0 e^{\mu t}$$

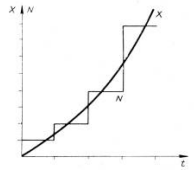
with the initial condition $X=X_0$ at $t=0$.

No living cell grows infinitely. Nutrients are absorbed through the surface of a cell, proportional to $r^2$ (a square of the cell's radius) while its volume grows as $r^3$. At some point the nutrient flow becomes insufficient to sustain the cell's growth. The cell either stops growing or divides.

Mean time between cell divisions $\tau$ can be calculated from:

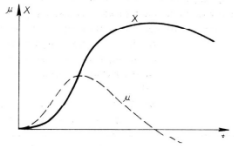
$$2X_0 = X_0 e^{\mu\tau}, \qquad \tau = \frac{\ln 2}{\mu}$$

If the cell divisions are independent, the concentration of biomass in a microbial culture is proportional to the number of cells. Under normal conditions the growth of cells is not synchronized (see figure below). Hence, the terms ***biomass growth*** and ***increase of the number of cells*** are equivalent.




***Bacterial growth: N – number of cells, X – biomass.***

The *per capita* growth rate $\mu$ is constant only if all components of the growth medium remain constant. Otherwise, it changes as a function of available resources (e.g. nutrients, oxygen, pH, light) and accumulated waste:




In 1845 Verhulst suggested the following model reflecting antagonistic interactions within a population:

$$\frac{dx}{dt} = \mu \cdot x - \beta \cdot x^2$$

Probability of death is proportional to the number of existing organisms. In the long-term limit the population stabilizes at $\boldsymbol{x_\infty = \mu/\beta}$.

Real populations do not realize constant *per capita* growth rates. By engineering the growth rate as a function of the population size, finely structures population models can be constructed.

If the growth rate decreases with increasing population size, then a finite limit, the ***carrying capacity***, is imposed on the population.

If the growth rate becomes negative for small population sizes, the population is be driven to ***extinction***.

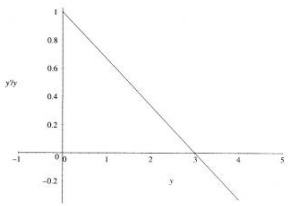
***Example: logistic growth***

When a biological population becomes too large, the *per capita* growth rate diminishes. This is because individuals interfere with each other and are forced to compete for limited resources. In the Verhulst model the *per capita* growth rate decreases linearly with population size ***y***:

$$\frac{1}{y}\frac{dy}{dt} = r\left(1 - \frac{y}{K}\right)$$

$$\frac{1}{y}\frac{dy}{dt} = r\left(1 - \frac{y}{K}\right)$$

This equation is known as the ***logistic equation***. The linearly decreasing *per capita* growth rate is depicted in the figure below.




The alternative form of this differential equation is:

$$\frac{dy}{dt} = ry\left(1 - \frac{y}{K}\right)$$

We see that the derivative is zero when:

$$y = 0$$

or

$$y = K$$

These are the stationary points of the equation. The stationary point ***y = K***, at which the *per capita* growth rate becomes zero, is called the ***carrying capacity*** (of the environment).

For small population sizes ***Y/K ≈ 0*** and the population increases exponentially with the *per capita* growth rate ***r***. Hence solutions are repelled from the stationary point ***y = 0***.

But as the population size approaches the carrying capacity ***K***, the growth rate decreases to zero and the population ceases to change in size.

Further, if the population size ever exceeds ***K***, the *per capita* growth rate becomes negative and the size decreases to ***K***. Hence, solutions are globally attracted to the stationary point ***y = K***.

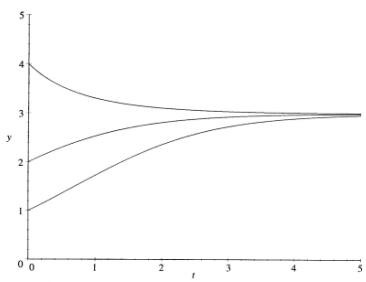
Rewriting the logistic equation as:

$$\frac{dy}{y\left(1-\frac{y}{K}\right)} = \left(\frac{1}{y} + \frac{\frac{1}{K}}{1-\frac{y}{K}}\right)dy = rdt$$

and integrating it directly, we obtain the explicit, analytical solution:

$$y(t) = \frac{y_0 K e^{rt}}{y_0\left(e^{rt}-1\right)+K}$$

A few of these solutions are shown in the figure below. Here, we take ***r=1***, ***K=3*** and find solutions with $y_0$***=1***, or ***2***, or ***4***.



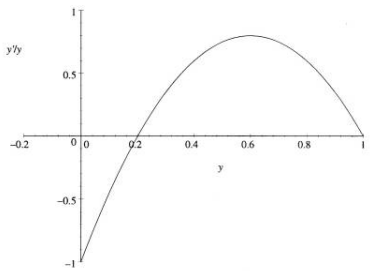

**Example: non-linear *per capita* growth rate**

Real populations are in danger of extinction if their size falls to a low level. Predation might eliminate the last few members completely, finding mates becomes more difficult, and a lack of genetic diversity renders the population susceptible to epidemics.

By constructing a *per capita* growth rate that is actually negative below some critical value, $\theta$, there results a population model that tends to extinction if the population size falls too low. For example:

$$\frac{1}{y}\frac{dy}{dt} = r\left(1-\frac{y}{K}\right)\left(\frac{y}{\theta}-1\right)$$

Where $0 < \theta < K$. The figure below shows the form of the *per capita* growth rate using the specific parameters: $r=1$, $\theta=1/5$, and $K=1$. This particular form is sometimes referred to as the ***predator pit***.



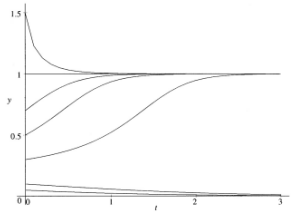

The stationary points of the equation:

$$\frac{1}{y}\frac{dy}{dt} = r\left(1-\frac{y}{K}\right)\left(\frac{y}{\theta}-1\right)$$

are $y = 0$, $y = \theta$ and $y = K$. But now $y = 0$ is asymptotically stable: if the starting value $y_0$ is near enough to $0$, then the solution tends to $0$ as $t$ increases. This follows because the sign of the rhs of the equation is negative for $0 < y < \theta$ causing $dy/dt < 0$. Hence $y$ will decrease.

On the other hand, a solution starting with $y_0 > \theta$ tends to $K$ as $t$ increases. This is because when $\theta < y < K$, the rhs of the equation is positive, so $dy/dt > 0$ also, hence $y$ will increase.

As before, solutions starting with $y_0 > K$ decrease asymptotically to $K$. Here are some solutions to the predator pit equation:




***Here is one more example of complex population modeling. The equation below represents a population model that engenders little growth for small populations, rapid for intermediate ones, and low growth again for large populations. This is achieved by the quadratic per capita growth rate:***

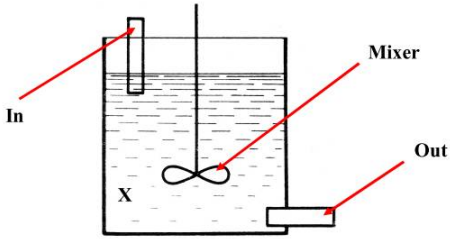
$$\frac{1}{y}\frac{dy}{dt} = ry\left(1-\frac{y}{K}\right)$$

**Croissance de la biomasse : chemostat**

Changes in conditions prevalent in growth media cause modifications of the *per capita* growth rate. However, even though the overall number of different factors affecting cellular growth may be high, usually only a ***small number of external parameters are relevant***. The regulation of the growth processes depends mainly on ***bottlenecks*** in the chain of chemical reactions.

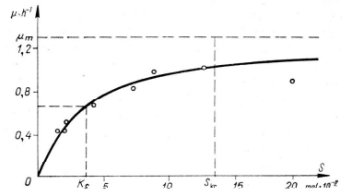
If one of the components is missing from the growth medium, cells grow significantly slower. The growth rate depends then directly on the concentration of the missing component. We talk about growing cells in a ***limiting medium***.

In order to study the characteristics of cellular growth, Monod (1942) proposed a ***flow cultivator*** (***homeostat***):




***The structure's volume V contains substrates S, biomass X and products P. Mixing assures homogeneity of the components. After a suitable time, a stable equilibrium is established.***

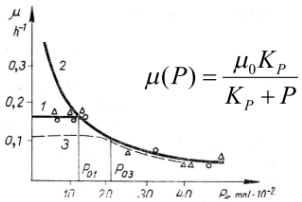
In homeostat, a stable equilibrium is established for given values of the flow rate and nutrient (i.e. substrate) concentration. The dependence of the *per capita* growth rate on the substrate ***S*** can be measured experimentally and has the following form:




$$\mu(S) = \frac{\mu_m S}{K_S + S}$$

***Note the similarity of the functional form of μ with the enzymatic reaction rates given by the Michaelis law.***

Influence of reaction products ***P*** (inhibitors) on cells can become a source of bottlenecks, significantly limiting cellular growth:




$$\mu(P) = \frac{\mu_0 K_P}{K_P + P}$$

The total effect (***S,P***) can be described by the following curve:

$$\mu(S,P) = \frac{\mu_m S K_P}{(K_S + S)(K_P + P)}$$

***Examples of homeostatic systems in living organisms:***

- **blood vessel system**

Bone marrow continuously produces erythrocytes (ca. $15 \cdot 10^6$/s), carried away by blood. Their concentration is constant (ca. $5 \cdot 10^6/mm^3$), hence they must continuously perish in liver and spleen.

- **renewal of fat tissue**

Studies based on isotopic markers have shown that the composition of fat tissue is continuously changing, while its total amount remains constant.

***Multicellular organisms function in the state of dynamic equilibrium.***

***Important for qualitative analysis of dynamic phenomena:***

To reduce the number of independent variables (i.e. the number of differential equations) to the absolute minimum (e.g. two, or three).

***The smaller the equation set the better the model!***

**In the case of homeostat we have the following options:**

| # of variables (and equations) | Cellular growth | Inhibition by product | Biological inertia |
|---|---|---|---|
| 2 | yes | no | no |
| 3 | yes | yes | no |
| 4 | yes | yes | yes |

**Variables:**

1) **Biomass concentration**
2) **Substrate concentration**
3) **Product concentration**
4) **Ribosome concentration**

In the simplest case of two variables (substrate, biomass), the equation describing the variation of **biomass** concentration ***x*** is composed of two terms:

$$\frac{dx}{dt} = \mu x - Dx$$

The first term describes the exponential growth of cells, while the second refers to the loss of biomass ***x*** due to steady flow of culture out of homeostat. The flow rate determines the ***dilution constant D***.

There are three components in the equation for the **substrate *y***. The nutrient is used by cells, hence its concentration decreases proportionally to the growth rate of the biomass, with the proportionality constant $\alpha$ corresponding to the efficiency of this process. It also flows out with the rate given by ***D***. However, its concentration increases because of the continuous supply with the initial concentration $y_0$. The balance is:

$$\frac{dy}{dt} = -\alpha\mu x + D(y_0 - y)$$

Taking into account the hyperbolic dependence of the *per capita* growth rate $\mu$ on the substrate concentration ***y*** we get:

$$\begin{cases} \dfrac{dx}{dt} = \dfrac{\mu_m y}{K_y + y} x - Dx \\ \dfrac{dy}{dt} = -\alpha \dfrac{\mu_m y}{K_y + y} x + D(y_0 - y) \end{cases}$$

In the **first stage of analysis** we will search for the stationary points of the system. By equating the derivatives to zero, we get two algebraic equations:

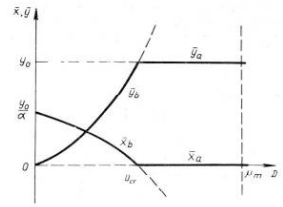
$$\left.\begin{aligned} \frac{d\bar{x}}{dt} &= 0 = \frac{\mu_m \bar{y}}{K_y + \bar{y}}\bar{x} - D\bar{x} \\ \frac{d\bar{y}}{dt} &= 0 = -\alpha \frac{\mu_m \bar{y}}{K_y + \bar{y}}\bar{x} + D(y_0 - \bar{y}) \end{aligned}\right\} \Rightarrow \left\{\begin{aligned} &\bar{x}\left(\frac{\mu_m \bar{y}}{K_y + \bar{y}} - D\right) = 0 \\ &\alpha\bar{x} + \bar{y} = y_0 \end{aligned}\right.$$

The second equation reflects a linear dependence of the two concentrations in the stationary state. From the first equation it follows that there are two solutions:

$$\bar{x} = \bar{x}_a = 0, \qquad \Rightarrow \qquad \bar{y} = \bar{y}_a = y_0$$

$$\bar{y} = \bar{y}_b = \frac{K_y D}{\mu_m - D}, \qquad \Rightarrow \qquad \bar{x} = \bar{x}_b = \frac{y_0 - \bar{y}_b}{\alpha} = \frac{\mu_m y_0 - D(K_y + y_0)}{\alpha(\mu_m - D)}$$

The two solutions (as functions of ***D***) are depicted in the figure below:




Note that $D < \mu_m$. For $D > D_{cr} = \mu_m y_0/(K_y+y_0)$ the biomass concentration becomes negative, which has no physical sense! In reality, it will remain zero (because of fast flow of biomass out of homeostat; growth no longer compensates the dilution). For $D < D_{cr}$ both solutions are acceptable.

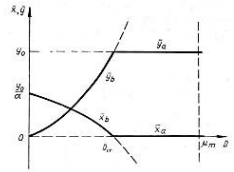
In the **second stage of analysis** we apply the perturbation theory to determine the stability of the two stationary points of the system. The analysis concerns the two following possibilities:

### 1. $D < D_{cr}$

Following the outline presented previously, we learn that for $D < D_{cr}$ the **trivial (zero) solution is unstable**. In other words, if $x(0) = 0$, no biomass can be grown. However, even a minimal number of cells can initiate the growth, which will continue until the **stable equilibrium (the second stationary point)** is reached. Subsequent small fluctuations will be compensated for, and the system can remain arbitrarily long in this stable state.

### 2. $D \geq D_{cr}$

For large flow rates, when $D \geq D_{cr}$, the situation is different. Placing cells in the homeostat has no consequences; the biomass will not develop. The flow directed to the outside of homeostat is so fast that dilution of biomass is instantaneous (before the cells multiply, they are carried out of the homeostat). In this case we can see that the stationary point corresponding to **zero concentration of biomass is stable**.



**Q**: ***Under what conditions does a homeostat work optimally?***

**A**: ***Define efficiency and optimize it!***

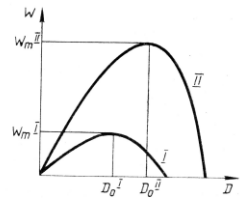
We define **efficiency** of a homeostat as the quantity of biomass it can produce per unit time per unit volume:

$$w = \frac{1}{V}\frac{\Delta m}{\Delta t} = \frac{1}{V}\frac{\Delta(xV)}{\Delta t} = \frac{\Delta x}{\Delta t} = \mu x$$

Hence, efficiency is defined as that part of the equation describing concentration of biomass, which corresponds to its growth. Moreover, in the state of stable equilibrium, this is equal to:

$$w = \mu x = Dx$$

As can be seen from the above, the efficiency tends to zero as $D \to 0$. Also, no biomass is produced when $D \geq D_{cr}$. Between the two limiting values there must occur a maximum of the efficiency, e.g. for $D = D_0$:




***Efficiency of a flow cultivator as a function of D***
***I: $y_0 = K_y$; II: $y_0 = 2K_y$***

The maximum of the efficiency:

$$w = D\bar{x} = D\frac{\mu_m y_0 - D(K_y + y_0)}{\alpha(\mu_m - D)}$$

can be found by equating its derivative $\partial w/\partial D$ to zero. Using the basic relationship:

$$\frac{d}{dx}\left(\frac{f(x)}{g(x)}\right) = \frac{f'(x)\cdot g(x) - f(x)\cdot g'(x)}{g^2(x)}$$

we arrive at the following expression:

$$\frac{\partial w}{\partial D} = \frac{[(\mu_m - D)(D_{cr} - 2D) + D(D_{cr} - D)](K_y + y_0)}{\alpha(\mu_m - D)^2} = 0$$

Since $(K_y + y_0) > 0$ and $(\mu_m - D) > 0$, we can write:

$$(\mu_m - D)(D_{cr} - 2D) + D(D_{cr} - D) = 0, \qquad D_{cr} = \frac{\mu_m y_0}{K_y + y_0}$$

$$D^2 - 2\mu_m D + \mu_m D_{cr} = 0$$

Of the two solutions available we keep the one for which $D < \mu_m$:

$$D_0 = \mu_m\left(1 - \sqrt{\frac{K_y}{K_y + y_0}}\right)$$

$$w_m = \frac{\mu_m}{\alpha}\left(\sqrt{K_y + y_0} - \sqrt{K_y}\right)$$

The second equation is the maximal efficiency, obtained for $D = D_0$.

The analysis accomplished so far has neglected the influence of inhibitors on the overall biomass production. Adding the third variable ***z*** (inhibitor concentration, usually waste product) and using the following form of the per capita growth rate:

$$\mu(y,z) = \frac{\mu_m y K_z}{(K_y + y)(K_z + z)}$$

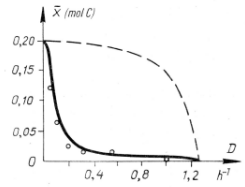
leads to the following equations for the stationary states of the system:

$$\bar{x} + \bar{y} + \bar{z} = y_0$$

$$\alpha\bar{x} + \bar{y} = y_0$$

$$\bar{x}\left[\frac{K_z \mu_m \bar{y}}{(K_y + \bar{y})(K_z + \bar{z})} - D\right] = 0$$

It can easily be found that $\boldsymbol{D_{cr}}$ (corresponding to the critical mass dilution) has the same value as before, but this time the curve describing the growth of biomass (stationary non-zero solution of ***x(D)***) has a different form:




***Solid curve and experimental points: effect of inhibition on biomass growth. Dashed curve: case with no inhibitor.***

It follows that in this case serious errors can result from predictions of biomass concentration based on the simplified model.

M2I BBS

# Modélisation et Simulation de Systèmes Biologiques

(#6 – mét. qualitatives)

**Georges Czaplicki, UPS / IPBS-CNRS**
**Tél. : 05.61.17.54.04, email : cgeorge@ipbs.fr**

**Eléments de la théorie qualitative de jeux des équations différentielles**

A natural assumption about the growth of a population is that ***the number of offspring at any given time is proportional to the number of adults at that time***. Let *y(t)* denote the number of adults at time *t*. In any given small interval of time $\Delta t$, the number of offspring in that time represents the change in the population $\Delta y$. The ratio $\Delta y/\Delta t$ is the average rate of growth of the population over the time period $\Delta t$. The derivative ***dy/dt*** is the instantaneous rate of growth at time ***t***. Thus, (with the assumption that new offspring are immediately adults) we obtain the following mathematical expression of the statement above:

$$\frac{dy}{dt} = ky$$

In other words, the rate of growth is proportional to the number present. The solution of this equation is given by an exponential curve:

$$y(t) = y_0 e^{kt}$$

This situation is typical and can be encountered nearly everywhere (populations, biomass, etc.).

Mathematical models of biological and biochemical processes are usually represented as ***systems of differential equations*** of the form:

$$\frac{dx_1}{dt} = F_1(x_1, x_2, ..., x_n)$$

.....

$$\frac{dx_n}{dt} = F_n(x_1, x_2, ..., x_n)$$

where functions $\boldsymbol{F_i}$ are in general nonlinear and do not depend explicitly on time ***t***, while $\boldsymbol{x_i}$ are unknown functions of time which may be constrained (e.g. concentrations cannot have negative values).

Systems of this kind are often encountered in classical description of motion (dynamics). Look at the second Newton's law (***F = ma***):

$$m\frac{d^2x}{dt^2} = F\left(x, \frac{dx}{dt}\right)$$

Upon substitution of a new variable ***y = dx/dt*** it becomes a system of two equations of the fist order:

$$\frac{dx}{dt} = y$$

$$\frac{dy}{dt} = \frac{1}{m}F(x, y)$$

One of the two independent variables is coordinate ***x***, the other is velocity ***y***. Coordinates and velocities fully describe a state (phase) of a system, therefore the plane in which we plot them is called a ***phase plane***. This term is used even if the functions $\boldsymbol{x_i}$ are of the same type (e.g. concentrations).

If $x_i$ are ***linear*** functions of their arguments, one can solve the system ***analytically***. In case of ***nonlinear*** equations, only ***numerical*** solutions are readily available. Their major drawback is that they do not permit to extrapolate the results to other regions of the phase plane (results of ***N*** consecutive calculations permit to determine the state, i.e. phase, of the system in ***N*** points of the phase plane, but do not give any basis to predict the result of calculation ***N+1***).

In most cases we are not interested in quantitative results. Sometimes it is even impossible to know the exact conditions of a given biological process (affinity values, initial conditions, etc.). Much more important is to be able to characterize the system qualitatively, to determine the existence of its ***stationary states***, the character of their ***stability***, and the changes in stability induced by modifications of model parameters.

The above problems constitute the essence of the ***qualitative theory of differential equations*** which, instead of finding the exact analysis of the solutions (i.e. no explicit form of functions $x_i$ is sought), focuses on ***general behavior of studied systems***, as deduced from the form of functions $F_i(x_1,\ldots,x_n)$.

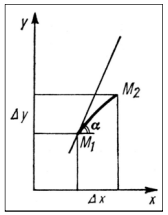
**The type of information provided by the theory:**

- **existence of stationary states (equilibrium)**
- **character of equilibrium (stable, unstable)**
- **phase trajectories (isoclines, asymptotic cycles)**

**Elements of analysis: phase portrait**

Many interesting mathematical models can be represented by a limited number of differential equations of the first order. We will consider the following example:

$$\frac{dx}{dt} = P(x,y)$$
$$\frac{dy}{dt} = Q(x,y)$$




At a given moment $t_1$ the system is in state $M_1$, characterized by $x(t_1)$ and $y(t_1)$. This state can be represented as a point in a two-dimensional phase plane (phase space in three dimensions, etc.). After time $\Delta t$ the coordinate $x$ will be incremented by $\Delta x$ and $y$ by $\Delta y$, as a result of which the system will be found in the state $M_2$. Analysis of infinitesimally small time increments permits to construct the ***trajectory*** of the system (see the figure below). Needless to say, $\tan(\alpha)$ (inclination of the straight, adjacent to the phase trajectory at a given point) is equal to the derivative $dy/dx$ at that point.

By eliminating time from the previous system of equations we arrive at the following single differential equation:

$$\frac{dy}{dx} = \frac{Q(x,y)}{P(x,y)}$$

The solution of this equation is a ***family of phase trajectories y=y(x,C)***, where the unknown parameter ***C*** can be determined from the initial conditions. Thus, we arrive at the ***phase portrait*** of the studied system.

***Note**: finding a solution of the above equation allows monitoring the relation between x and y, but their explicit time variations x(t) and y(t) still remain unknown. The role of the qualitative theory of dynamic systems is to understand the properties of the functions x(t) and y(t) from the analysis of the studied system's phase portrait.*

According to the Cauchy theorem, each point of the phase plane can be traversed by only ***one*** trajectory. The only exceptions are ***singular points***, at which the inclination of the adjacent straight, tan(α), is not defined:

$$\frac{dy}{dx} = \frac{0}{0}$$
$$\Downarrow$$
$$P(x,y) = 0 \quad \text{AND} \quad Q(x,y) = 0$$

Multiple trajectories may traverse the singular points of the phase plane.

The conditions ***P(x,y)=0*** and ***Q(x,y)=0*** are equivalent to setting the rates of time variation of ***x*** and ***y*** to 0: ***dx/dt = 0*** and ***dy/dt = 0***. Hence, the singular points of the phase plane correspond to ***stationary states*** of the studied dynamic system.

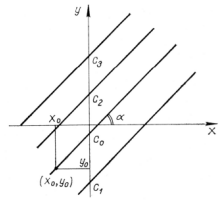
**Example:**

$$\frac{dx}{dt} = A = const \qquad \frac{dy}{dt} = B = const$$

There are no singular points in the phase plane. Elimination of time gives the following equation:

$$\boxed{\frac{dy}{dx} = \frac{B}{A}} \quad \text{whose solution is given by:} \quad \boxed{y = \frac{B}{A}x + C}$$

which defines a family of straight lines (see figure below), with ***tan(α)=B/A***. ***C*** depends on the initial conditions [e.g. $C_0 = y_0 - (B/A)x_0$].



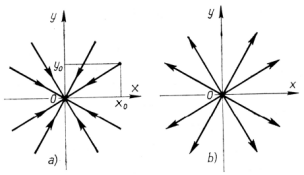

The direction of the movement along the lines can only be determined from the analysis of the original equations ({A, B} > 0 => upwards, {A, B} < 0 => downwards).

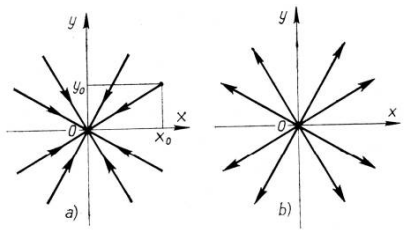
**Example :** $\frac{dx}{dt} = -x \qquad \frac{dy}{dt} = -y$

There is one singular point here, given by the coordinates: ***x*** = ***y*** = 0.
The family of trajectories is given by the equation (see figure below):

$$\boxed{\begin{array}{ll} dx = -xdt & \mathrm{dy} = -ydt \\ \frac{dy}{dx} = \frac{y}{x} & \end{array}}$$

$$\begin{array}{ll} dx = -xdt & \mathrm{dy} = -ydt \\ \frac{dy}{dx} = \frac{y}{x} & \frac{dy}{y} = \frac{dx}{x} \\ \int \frac{dy}{y} = \int \frac{dx}{x} & \ln(y) = \ln(x) + const = \ln(Cx) \\ y = Cx & \left( C = \frac{y_0}{x_0} \right) \end{array}$$




Vectors in ***(a)*** show the system's time evolution, which depends on the signs in the rhs of the above equation. Since they are negative, all trajectories converge to the singular point (0,0) representing a ***stable node***.

If the signs were both positive, the trajectories would diverge to infinity, as in ***(b)***. If the system would initially be at (0,0), any random fluctuation would drive it away from it and the system would drift away along one of the trajectories. In this case the singular point (0,0) is an ***unstable node***.

## Elements of analysis: singular points

**Q:** ***How to determine in a general way the type of stability of a singular point in the phase plane?***

**A:** Apply a ***small perturbation*** to check if the system returns to the initial point.

Let's begin with the original set of equations:

$$\frac{dx}{dt} = P(x, y)$$
$$\frac{dy}{dt} = Q(x, y)$$

Let ***($x_s$, $y_s$)*** denote the coordinates of a singular point in the phase plane of the system. Application of a small perturbation will drive the system away from the stationary point (equilibrium). The coordinates become:

$$x = x_s + \xi, \qquad y = y_s + \eta \qquad \left( \left| \frac{\xi}{x_s} \right| << 1, \quad \left| \frac{\eta}{y_s} \right| << 1 \right)$$

We substitute the latter into the former, expanding the functions ***P(x,y)*** and ***Q(x,y)*** into a Taylor series about the singular point ***($x_s$, $y_s$)***. Since the perturbation is small, we may limit the expansion to first order terms only.

The Taylor expansion yields:

$$P(x, y) = P(x_s, y_s) + a_{11}\xi + a_{12}\eta + \ldots$$
$$Q(x, y) = Q(x_s, y_s) + a_{21}\xi + a_{22}\eta + \ldots$$

The coefficients $\boldsymbol{a_{ij}}$ are partial derivatives of functions ***P*** and ***Q*** calculated at the singular point ***($x_s$, $y_s$)***:

$$a_{11} = \left.\frac{\partial P}{\partial x}\right|_{x_s, y_s} \qquad a_{12} = \left.\frac{\partial P}{\partial y}\right|_{x_s, y_s}$$

$$a_{21} = \left.\frac{\partial Q}{\partial x}\right|_{x_s, y_s} \qquad a_{22} = \left.\frac{\partial Q}{\partial y}\right|_{x_s, y_s}$$

On the other hand, we know that at the singular point ***($x_s$, $y_s$)***:

$$P(x_s, y_s) \equiv 0, \qquad Q(x_s, y_s) \equiv 0.$$

Hence, in the first-order approximation, we get the following relations for the small perturbations about the singular point:

$$\frac{dx}{dt} = P \quad \Rightarrow \quad \frac{d}{dt}(x_s + \xi) = P \quad \Rightarrow \quad \frac{d\xi}{dt} = a_{11}\xi + a_{12}\eta$$
$$\frac{dy}{dt} = Q \quad \Rightarrow \quad \frac{d}{dt}(y_s + \eta) = Q \quad \Rightarrow \quad \frac{d\eta}{dt} = a_{21}\xi + a_{22}\eta$$

The above ***linearized*** equations describe the system's behavior near the singular point. This method of analysis is called ***perturbation theory***.

In ***matrix notation*** the equations can be written in a compact way:

$$\left.\begin{aligned}\frac{d\xi}{dt} &= a_{11}\xi + a_{12}\eta \\ \frac{d\eta}{dt} &= a_{21}\xi + a_{22}\eta\end{aligned}\right\} \quad \Rightarrow \quad \frac{d\mathbf{v}}{dt} = \mathbf{A}\mathbf{v}, \qquad \mathbf{v} = \begin{pmatrix}\xi \\ \eta\end{pmatrix} \quad \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22}\end{pmatrix}$$

A solution is sought as a function of the form:

$$\xi = Ae^{\lambda t}, \qquad \eta = Be^{\lambda t}$$

Substituting these expressions into the above equations and dropping the exponential term yields:

$$\lambda A = a_{11}A + a_{12}B$$
$$\lambda B = a_{21}A + a_{22}B$$

Solving the second equation for ***A*** [***A = B(λ – a₂₂)/a₂₁***] and substituting it into the first one gives:

$$\left[(a_{11} - \lambda)\cdot(a_{22} - \lambda) - a_{12}\cdot a_{21}\right]\cdot B = 0$$

Since only non-zero amplitudes are interesting (***B ≠ 0***), we get:

$$\lambda^2 - (a_{11} + a_{22})\lambda - a_{12}\cdot a_{21} + a_{11}a_{22} = 0$$

$$\left.\begin{aligned}2\delta &= -(a_{11} + a_{22}) \\ \omega_0^2 &= a_{11}a_{22} - a_{12}a_{21}\end{aligned}\right\} \quad \Rightarrow \quad \lambda^2 + 2\delta\lambda + \omega_0^2 = 0$$

**Note**: ***identical expression can be obtained from matrix A by calculating:***

$$\det\left|\mathbf{A} - \lambda\mathbf{I}\right| = 0$$

Another way to obtain the characteristic equation is to convert two equations of the first order into one equation of the second order:

$$\left.\begin{aligned}(a)\quad \frac{d\xi}{dt} &= a_{11}\xi + a_{12}\eta \\ (b)\quad \frac{d\eta}{dt} &= a_{21}\xi + a_{22}\eta\end{aligned}\right\} \quad \Rightarrow \quad \left\{\begin{aligned}&\frac{d^2\xi}{dt^2} = a_{11}\frac{d\xi}{dt} + a_{12}\frac{d\eta}{dt}, \quad \text{from (b)}: \\ &\frac{d^2\xi}{dt^2} = \left(a_{11}{}^2 + a_{12}a_{21}\right)\xi + \left(a_{11}a_{12} + a_{12}a_{22}\right)\eta, \quad \text{from (a)}: \\ &\frac{d^2\xi}{dt^2} - (a_{11} + a_{22})\frac{d\xi}{dt} + (a_{11}a_{22} - a_{12}a_{21})\xi = 0\end{aligned}\right.$$

The last equation can be written as:

$$\left.\begin{aligned}2\delta &= -(a_{11} + a_{22}) \\ \omega_0^2 &= a_{11}a_{22} - a_{12}a_{21}\end{aligned}\right\} \quad \Rightarrow \quad \frac{d^2\xi}{dt^2} + 2\delta\frac{d\xi}{dt} + \omega_0^2\xi = 0$$

The solution is sought as a function of the form:

$$\xi(t) = Ae^{\lambda t}$$

The derivatives of this function are given by:

$$\frac{d\xi(t)}{dt} = A\lambda e^{\lambda t} = \lambda\xi(t)$$
$$\frac{d^2\xi(t)}{dt^2} = \lambda\frac{d\xi(t)}{dt} = \lambda^2\xi(t)$$

Substitution of the above relations into the equation for ***ξ(t)*** yields:

$$\left(\lambda^2 + 2\delta\lambda + \omega_0^2\right)\xi(t) = 0$$

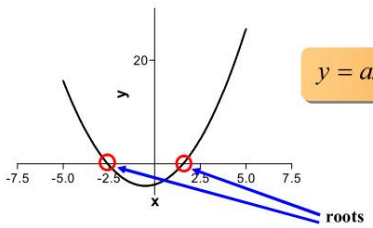
which, for non-zero ***ξ(t)***, gives the characteristic equation for ***λ***:

$$\lambda^2 + 2\delta\lambda + \omega_0^2 = 0$$

## Math Reminder: properties of a parabole

### Definition of the curve

A parabole is a curve, given by the equation:




$$y = ax^2 + bx + c$$

By replacing ***x*** by a new variable ***x = Az + B*** we obtain:

$$\begin{aligned} y &= a(Az+B)^2 + b(Az+B) + c \\ &= aA^2z^2 + (2aAB + bA)z + (aB^2 + bB + c) \end{aligned}$$

We choose the constants ***A*** & ***B*** in such a way as to simplify the above expression:

$$\left.\begin{aligned} aA^2 &= 1 \\ 2aAB + bA &= 0 \end{aligned}\right\} \quad \Rightarrow \quad \left\{\begin{aligned} A &= a^{-\frac{1}{2}} \\ B &= -\frac{b}{2a} \end{aligned}\right\} \quad \Rightarrow \quad x = \frac{z}{\sqrt{a}} - \frac{b}{2a}$$

As a result, we get the following equivalent equation:

$$y = z^2 - \frac{\Delta}{4a}, \qquad \Delta = b^2 - 4ac$$

### How to find roots of a quadratic equation?

To find roots of the equation:

$$y = z^2 - \alpha^2, \qquad \alpha = \frac{b^2 - 4ac}{4a}$$

we search for ***z*** such that:

$$0 = z^2 - \alpha^2$$

Since the following relation holds:

$$(a+b)\cdot(a-b) = a^2 - ab + ab + b^2 = a^2 - b^2$$

we may write:

$$0 = (z-\alpha)\cdot(z+\alpha)$$

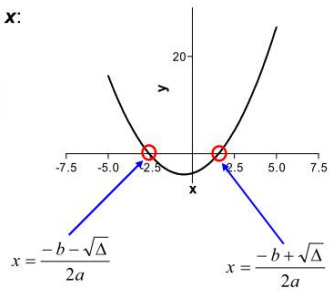
$$0 = (z-\alpha)\cdot(z+\alpha)$$

which has two obvious roots: ***z = +α*** and ***z = -α***. Hence, the solution is:

$$z = \mp\alpha = \pm\sqrt{\frac{\Delta}{4a}}$$

Finally, we return to the variable ***x***:

$$x = \frac{z}{\sqrt{a}} - \frac{b}{2a} = \frac{-b \mp \sqrt{\Delta}}{2a}$$



## Analysis of roots

Obviously, the values of ***x*** depend on parameters ***a***, ***b*** and ***c***:

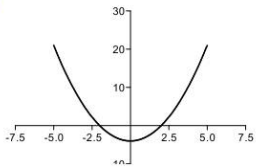
$$x = \frac{-b \mp \sqrt{\Delta}}{2a} = \frac{-b \mp \sqrt{b^2 - 4ac}}{2a}$$

***There are three possibilities:***

**1. $\Delta > 0$**

There are two different real roots.

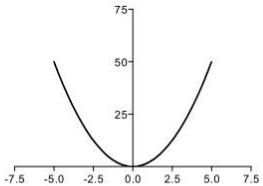
$$x_1 = \frac{-b - \sqrt{\Delta}}{2a}$$
$$x_2 = \frac{-b + \sqrt{\Delta}}{2a}$$




**2. $\Delta = 0$**

There is one double real root.

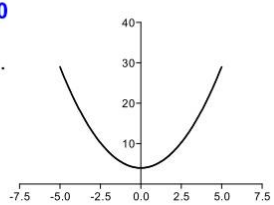
$$x_1 = x_2 = \frac{-b}{2a}$$




**3. $\Delta < 0$**

There are two complex conjugate roots.

$$x_1 = \frac{-b - i\sqrt{|\Delta|}}{2a}$$
$$x_2 = \frac{-b + i\sqrt{|\Delta|}}{2a}$$




$$\boxed{\lambda^2 + 2\delta\lambda + \omega_0^2 = 0}$$

The characteristic equation has two roots, given by $\lambda_1$ and $\lambda_2$:

$$\lambda_{1,2} = -\delta \mp \omega$$
$$\omega^2 = \delta^2 - \omega_0^2$$

In the case of multiple roots, the general solution of the equations for $\xi$ and $\eta$ is a linear superposition of specific solutions, corresponding to each of the roots:

$$\xi = C_{11}e^{\lambda_1 t} + C_{12}e^{\lambda_2 t}$$
$$\eta = C_{21}e^{\lambda_1 t} + C_{22}e^{\lambda_2 t}$$

Amplitudes $C_{ij}$ depend on initial conditions (only two of the four are independent).

**Values of $\lambda_1$ and $\lambda_2$ determine the system behavior about the singular point and thus characterize the nature of the singularity.**

**Analysis of the nature of the stationary point allows drawing conclusions concerning the stability of the system near this particular point.**

In order to further the analysis, various combinations of $\lambda_1$ and $\lambda_2$ values have to be considered in detail. Their character and values depend on the sign of the determinant ***D***:

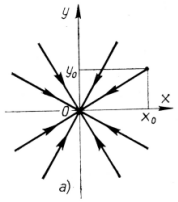
$$\begin{aligned} D &= (a_{11} + a_{22})^2 + 4(a_{12}a_{21} - a_{11}a_{22}) \\ &= (a_{11} - a_{22})^2 + 4a_{12}a_{21} \\ &= 4(\delta^2 - \omega_0^2) \\ &= 4\omega^2 \end{aligned}$$

**First case**: D ≥ 0

In this case both $\lambda_1$ and $\lambda_2$ are real. There are three possibilities here:

**1. Both $\lambda_1$ and $\lambda_2$ are negative.**

Solutions for $\xi$ and $\eta$ (i.e. evolutions of perturbations about the stationary point $(x_s, y_s)$) are combinations of exponentially decaying functions. Hence, the singular point describes a ***stable equilibrium*** (see figure ***a)*** below).

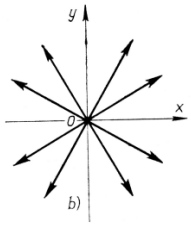



**stable node**

**First case**: D ≥ 0

**2. Both $\lambda_1$ and $\lambda_2$ are positive.**

Both solutions (perturbations) exponentially increase, driving the system away from the stationary point. This is an ***unstable equilibrium*** (see ***b)*** below).




**unstable node**

**First case**: D ≥ 0

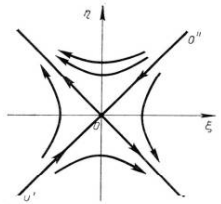
***3.* $\lambda_1$ and $\lambda_2$ have opposite signs.**

If the roots have mixed signs (e.g. $\lambda_1 > 0$ and $\lambda_2 < 0$), the singular point is ***unstable***, because the term with the positive exponent will dominate after a sufficiently long time.

An ***exception*** may occur only when the initial conditions are such that the coefficients $C_{ij}$ for terms with positive exponents are zero, so that only decreasing terms are left, e.g.:

$$\xi = C_{12}e^{-|\lambda_2|t}, \qquad \eta = C_{22}e^{-|\lambda_2|t}$$

As a consequence, we observe a straight line in the phase plane (***O'-O''*** in the figure below), given by $\xi = (C_{12}/C_{22})\eta$, along which the system converges to the stationary point (***O***). The other straight line passing through the singular point is the trajectory that diverges to infinity. The two straight lines are called ***separators*** and divide the phase plane into regions in which trajectories have common features.




Other trajectories feature an interesting property: initially they approach the singular point, but later on they go away from it. Such a point is called ***saddle*** and is analogous to what is known in geography as ***pass***: it's the lowest point between two peaks.

**Second case**: D < 0

$$D = 4\left(\delta^2 - \omega_0^2\right) = 4\omega^2 < 0 \quad \Rightarrow \quad \delta^2 < \omega_0^2$$

In this case the roots of the characteristic equation are complex conjugate numbers:

$$\lambda_{1,2} = -\delta \pm i\omega, \qquad \omega^2 = \omega_0^2 - \delta^2$$

The exponential term in the solution takes the form:

$$e^{\lambda t} = e^{-\delta t \pm i\omega t} = e^{-\delta t} \cdot e^{\pm i\omega t} = e^{-\delta t}\left(\cos\omega t \pm i \sin\omega t\right)$$

The general solution of the equations is now given by the expression:

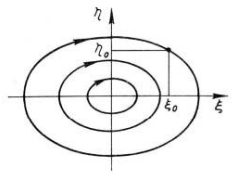
$$\xi(t) = e^{-\delta t}\left(C_1 \cos\omega t \pm C_2 \sin\omega t\right)$$

In view of the periodicity of the term on the right (containing $\omega$) it can be seen that the character of the singular point is determined exclusively by the value and the sign of $\delta$. There are three cases possible:

**Second case**: D < 0

**1. $\delta = 0$.**

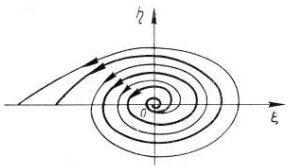
***ξ(t)*** is a periodic function of time, i.e. a perturbation causes non-decaying oscillations of the system with frequency $\omega$. The trajectories in the phase plane are concentric ellipses whose shape depends on the initial conditions. Singularity of this type is called a ***center***. The stationary point is ***neutrally stable***.




**Second case**: D < 0

**2. $\delta < 0$.**

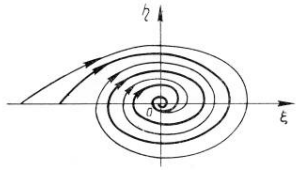
The exponential factor $e^{|\delta|t}$ drives the system away from the stationary point to infinity. The trajectories represent a family of unwinding spirals (see figure below). The stationary point is an ***unstable manifold***.




**Second case**: D < 0

**3. $\delta > 0$.**

The perturbation causes decaying oscillations and return to the stationary point, called a ***stable manifold***.




***Note 1***: *the density of the spiral coils depends on the ratio of $\delta$ and $\omega$.*

***Note 2***: ***Center*** *($\delta$ = 0) is* ***unstable****, because a fluctuation leading to a small modification of $\delta$ leads to its transformation into a* ***stable*** *($\delta$ > 0) or* ***unstable*** *($\delta$ < 0)* ***manifold****. Hence, center is on the border between these two different physical states of the system. Such a radical change of the state caused by a small perturbation is called* ***bifurcation****.*

## Elements of analysis: asymptotic limits

**Q**: ***What happens ultimately to the trajectories, which diverge from the unstable stationary points?***

**A**: ***Learn to construct a phase portrait of the system far from equilibrium.***

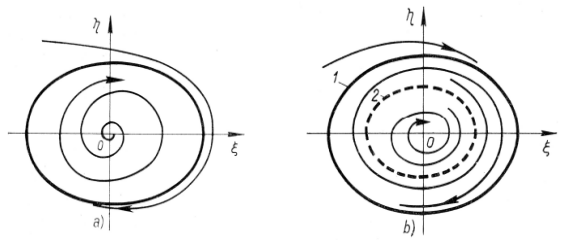
The theory of perturbations allows studying the behavior of a system in the proximity of the stationary points. When the system leaves an unstable equilibrium, its trajectories lead to points located so far from the initial stationary point that the linearized equations no longer apply. Further analysis can be done by ***qualitative geometric construction of integration curves***.

**Note**: ***In reality, trajectories should not diverge to infinity, because in the physical world parameter values (concentrations, velocities, etc.) cannot be infinite. If an analysis of a system of differential equations indicates existence of a stable stationary point in infinity, it usually means that the model portraying that system is imperfect, e.g. some limiting factor has been left out.***

Trajectories drifting away from unstable stationary points tend to attain their ***basins of attraction***. There are two possibilities:

1. *Somewhere near the unstable stationary point there exists a **stable equilibrium**, to which the trajectories converge.*
2. *There are no stable equilibria nearby, but trajectories do not diverge to infinity. Instead, they gradually approach a **closed curve**, which represents an **asymptotic cycle**.*

**Example**: a system featuring one ***(a)*** and two ***(b)*** ***asymptotic cycles***. The singular point is at the origin ***O*** of the coordinate system.

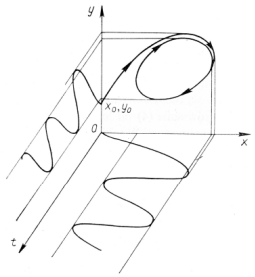



In ***(a)*** the trajectories leave an unstable stationary point ***O*** to reach the stable asymptotic cycle (thick line).

In ***(b)*** the trajectories diverge from the unstable asymptotic cycle ***2*** (dotted line) either to the stable stationary point ***O*** or to the stable asymptotic cycle ***1***.

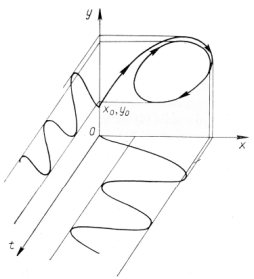
**Q**: ***What physical state corresponds to an asymptotic cycle?***

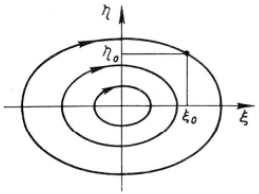
**A**: Looking at a system's 3D phase portrait (where time ***t*** has been added to the 'regular' coordinates ***x*** and ***y***) facilitates the analysis of a closed cycle (see figure below). Since an asymptotic cycle is a closed curve, periodic movement will be observed. Thus, an asymptotic trajectory corresponds to ***oscillations with constant amplitude***. It is a self-regulating system in the sense that oscillations occur spontaneously, without any external periodic factors, and because the system is stabilized in that state.



**Q**: ***What is the difference between an asymptotic cycle and a center?***

**A**: ***In both cases the system moves along elliptical trajectories. However, in the case of center the amplitude depends on initial conditions and is very sensitive to small perturbations.***




## Elements of analysis: isoclines

**Definition**: ***isoclines*** are such curves in the phase plane, which intersect all integration curves (solution trajectories) always at the same angle. Hence, they satisfy the condition ***dy/dx = const***. Particularly important are ***principal isoclines***, which satisfy the conditions:

$$\frac{dy}{dx} = 0, \qquad \frac{dy}{dx} = \infty$$

The equation we're trying to solve is given by the relation:

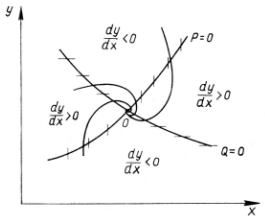
$$\frac{dy}{dx} = \frac{Q(x,y)}{P(x,y)}$$

Hence, the principal isoclines are determined by the expressions:

$$Q(x,y) = 0$$
$$P(x,y) = 0$$

**Singular points are located at the intersection of these curves.**

The figure below illustrates the fact that by plotting principal isoclines and keeping track of the signs of derivatives we can sketch trajectories, which approach the stationary point ***0***.



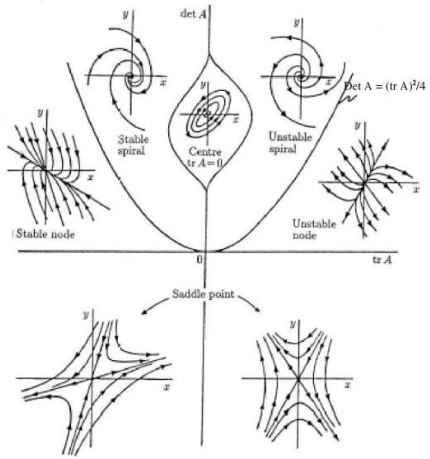
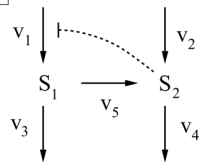




FIG. 5.5 – Portraits de Phases dans le plan $\det A - \mathrm{tr} A$

## Exemple




$$\frac{d}{dt}s_1(t) = \frac{k_1}{1+(s_2(t)/K)^n} - k_3 s_1(t) - k_5 s_1(t)$$
$$\frac{d}{dt}s_2(t) = k_2 + k_5 s_1(t) - k_4 s_2(t).$$

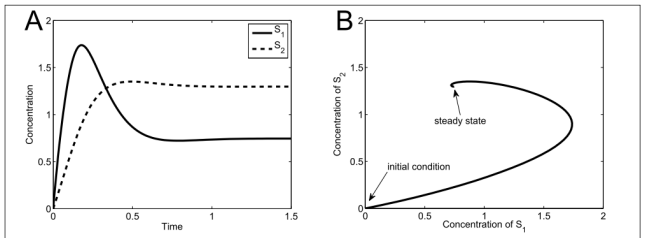



Figure 4.2: Simulation of model (4.1). **A.** Concentrations plotted against time. Both $[S_1]$ and $[S_2]$ overshoot their steady state values before coming to rest. **B.** Concentration $[S_1]$ plotted against concentration $[S_2]$ in the phase plane. Parameter values: (in concentration $\cdot$ time$^{-1}$) $k_1 = 20$ and $k_2 = 5$; (in concentration) $K = 1$; (in time$^{-1}$) $k_3 = 5$, $k_4 = 5$, and $k_5 = 2$; and $n = 4$. Units are arbitrary.

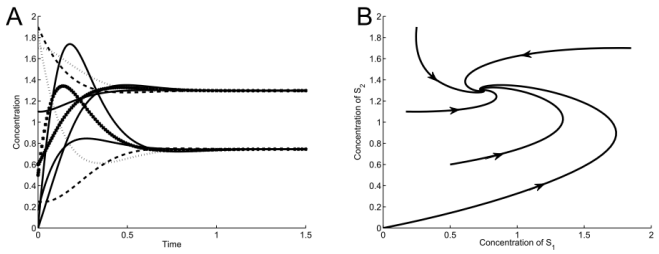



Figure 4.3: Simulations of model (4.1). **A.** Multiple time-courses confirm the steady state concentrations, but the transient behaviour cannot be usefully resolved. **B.** On the phase plane, the individual trajectories provide a unified picture of the dynamic behaviour of the system. Parameters as in Figure 4.2.

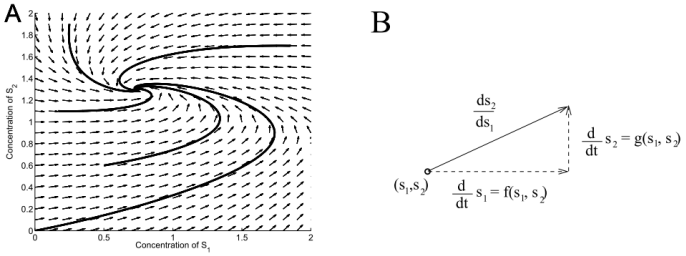



Figure 4.4: Direction field for model (4.1). **A.** The field of arrows indicates the direction of motion at each point. Trajectories are curves that lie tangent to the arrows; they follow the flow. **B.** Arrows in the direction field can be generated directly from the model—no simulation is needed. At each point $(s_1, s_2)$, the direction of the arrow is the slope of $s_2$ with respect to $s_1$ ($\frac{ds_2}{ds_1}$). This slope can be determined from the model dynamics, which specify the rates of change of $s_2$ and $s_1$. In Panel A, these vectors have been normalized to display a field of arrows of equal length.

## Isoclines

$$s_1 = \frac{k_1}{(k_3+k_5)(1+(s_2/K)^n)}$$
$$s_2 = \frac{k_2 + k_5 s_1}{k_4}.$$

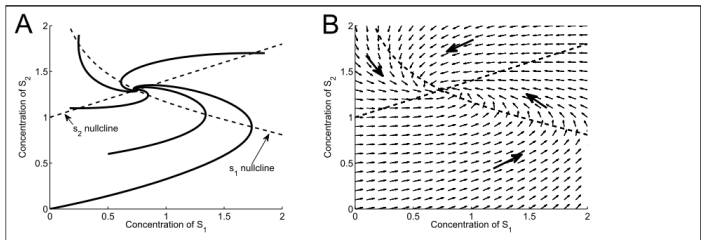



Figure 4.5: Nullclines for model (4.1). **A.** The trajectories intersect the nullclines at turning points. The steady state occurs at the intersection of the nullclines. **B.** The nullclines divide the phase plane into four regions. Since the direction arrows only 'flip' directions at the nullclines, each region is characterized by motion in a particular direction (up/down, left/right), as indicated.

## Bifurcation

**Exercise 4.4.1** Consider the differential equation

$$\frac{d}{dt}x(t) = (a-1)x(t).$$

By determining the sign of the rate of change $\frac{dx}{dt}$ for positive and negative values of $x$, verify that the steady state at $x = 0$ is stable if $a < 1$, and unstable if $a > 1$. The parameter value $a = 1$ is thus a bifurcation point for this system. □

$$\frac{dx}{x} = (a-1)dt$$

$$\int \frac{dx}{x} = (a-1)\int dt$$

$$\ln x = (a-1)t + const$$

$$x = Ce^{(a-1)t}$$

$$\frac{dx}{dt} = (a-1)x > 0 \quad \rightarrow \quad a > 1$$

$$\frac{dx}{dt} = (a-1)x < 0 \quad \rightarrow \quad a < 1$$

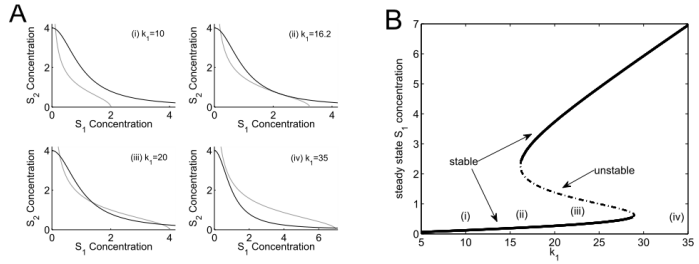



Figure 4.19: Bifurcation diagram for model (4.2). **A.** Nullclines at various values of $k_1$. As $k_1$ increases, the $s_1$-nullcline (gray curve) shifts: (i) at low $k_1$ there is a single steady state (low $[S_1]$, high $[S_2]$); (ii) at a higher value of $k_1$, a new steady state appears when a new intersection appears; (iii) at still higher values, three intersection points are exhibited—the system is bistable; (iv) finally, at high $k_1$ values, there is again a single intersection point (high $[S_1]$, low $[S_2]$). **B.** Bifurcation diagram showing the $S_1$ steady state concentration as a function of the value of parameter $k_1$. At low and high $k_1$ values, the system is monostable and exhibits a single stable steady state (solid curves). Over a mid-range interval, the two stable steady states co-exist, separated by an unstable steady state (dashed curve). The $k_1$ values at which steady states appear or disappear are saddle-node bifurcations. The $k_1$ values represented in Panel A are indicated. Parameter values: $k_2 = 20$ (concentration $\cdot$ time$^{-1}$), $K_1 = K_2 = 1$ (concentration), $k_3 = k_4 = 5$ (time$^{-1}$), $n_1 = n_2 = 2$. Units are arbitrary.

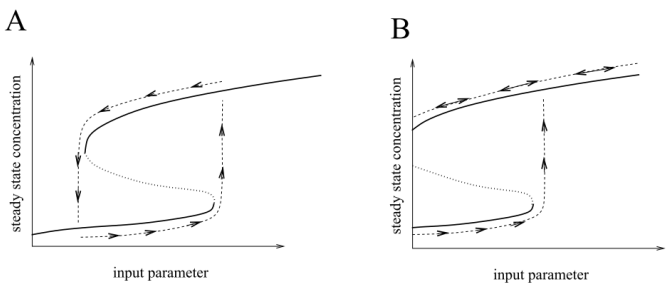



Figure 4.20: Switching in bistable systems. **A.** Hysteresis loop. Changes in the input parameter can push the system from one steady state to the other. Over the intermediate bistable region, the state depends on the recent past. Transitions between the two states occur abruptly at the bifurcation points. **B.** Irreversible switching. If one of the two bifurcation points is inaccessible, the system can become trapped in one of the steady states.

## Limit cycle oscillations

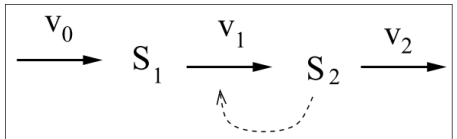



Figure 4.14: Autocatalytic biochemical reaction network.

We model the network as

$$\begin{aligned} \frac{d}{dt}s_1(t) &= k_0 - k_1[1 + (s_2(t)/K)^n]s_1(t) \\ \frac{d}{dt}s_2(t) &= k_1[1 + (s_2(t)/K)^n]s_1(t) - k_2 s_2, \end{aligned} \tag{4.10}$$

The allosteric activation is presumed to be strongly cooperative.

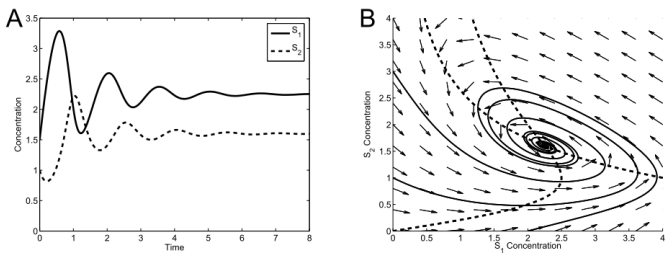



Figure 4.15: Model (4.10) with moderate nonlinearity. **A.** Time series. The species concentrations exhibit damped oscillations as they converge to steady state. **B.** Phase plane. Damped oscillations correspond to trajectories (solid curves) spiraling toward the steady state at the intersection of the nullclines (dashed curves). Parameter values: $k_0 = 8$ (concentration $\cdot$ time$^{-1}$), $k_1 = 1$ (time$^{-1}$), $K = 1$ (concentration), $k_2 = 5$ (time$^{-1}$) and $n = 2$. Units are arbitrary.

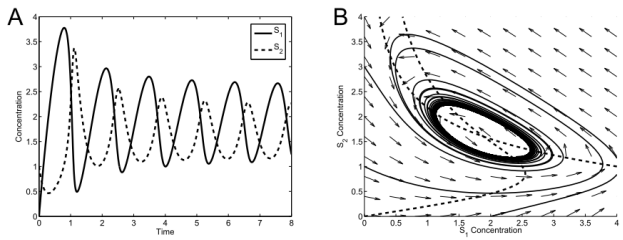



Figure 4.16: Model (4.10) with strong nonlinearity. **A.** This time series shows convergence to sustained periodic behaviour. **B.** In this phase portrait all trajectories converge to a cyclic track called a limit cycle. The steady state at the intersection of the nullclines is an unstable spiral point. Parameter values as in Figure 4.15 except $n = 2.5$.

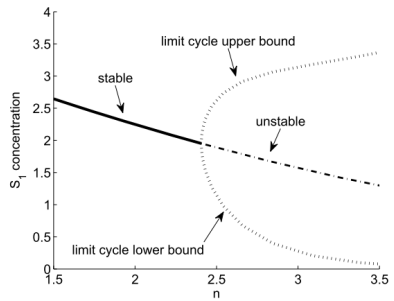



Figure 4.21: Bifurcation diagram for the autocatalytic model (4.10). For small $n$ values, a single stable steady state is shown. At higher $n$ values, this steady state is unstable. At the bifurcation point ($n = 2.4$) a stable limit cycle is born; the two dotted curves show the maximal and minimal concentrations reached by the limit cycle. The transition point is called a Hopf bifurcation. Parameter values as in Figure 4.15.

M2I BBS

# Modélisation et Simulation de Systèmes Biologiques

(#7 – proc. périod.)

Georges Czaplicki, UPS / IPBS-CNRS
Tél. : 05.61.17.54.04, email : cgeorge@ipbs.fr

## Processus périodiques

A periodic kinetic equation has been described for the first time by Lotka (1925). A purely hypothetical reaction serves as a didactic model for studying periodic processes. It is given by the following schema:

$$A \xrightarrow{k_0} X \xrightarrow{k_1} \overleftarrow{Y} \xrightarrow{k_2} B$$

The molecules of the substrate ***A***, which is in excess, convert to the substance ***X*** with a constant rate $k_0$ (reaction of the zeroth order). ***X*** is converted to ***Y*** in a second order reaction (the reaction rate $k_1$ depends on the concentrations of both ***X*** and ***Y***, which is denoted by a reverse arrow above ***Y***). ***Y*** undergoes an irreversible decay, forming product ***B*** with the rate $k_2$.

$$\boxed{A \xrightarrow{k_0} X \xrightarrow{k_1} \overleftarrow{Y} \xrightarrow{k_2} B}$$

The mathematical model of this system is described by the following set of equations:

$$\frac{dX}{dt} = k_0 - k_1 XY$$
$$\frac{dY}{dt} = k_1 XY - k_2 Y$$
$$\frac{dB}{dt} = k_2 Y$$

The first two equations do not depend on ***B***, hence they can be analyzed independently of the third one. We shall proceed as usual, beginning with the determination of stationary points in the phase plane of the studied system, which are obtained by solving the algebraic equations, resulting from equation all derivatives to zero.

The equations:

$$0 = k_0 - k_1 \overline{X}\overline{Y}$$
$$0 = k_1 \overline{X}\overline{Y} - k_2 \overline{Y}$$

have two solutions. The trivial one (***Y=0***) is not interesting and will not be discussed. The other one is:

$$\overline{X} = \frac{k_2}{k_1}$$
$$\overline{Y} = \frac{k_0}{k_2}$$

To analyze the type of stability associated with this stationary point we will introduce small perturbations x(t) and y(t) about the equilibrium:

$$X(t) = \overline{X} + x(t)$$
$$Y(t) = \overline{Y} + y(t)$$

Substituting the above relations to the original equations we obtain the expressions for the perturbations:

$$\frac{dx}{dt} = -\frac{k_0 k_1}{k_2} x - k_2 y - k_1 xy$$
$$\frac{dy}{dt} = \frac{k_0 k_1}{k_2} x + k_1 xy$$

Neglecting small terms of the second order (***xy***) we get a linearized system of equations:

$$\frac{dx}{dt} = -\frac{k_0 k_1}{k_2} x - k_2 y$$
$$\frac{dy}{dt} = \frac{k_0 k_1}{k_2} x$$

Introducing the following denotations:

$$\frac{k_0 k_1}{k_2} = 2\delta, \qquad k_0 k_1 = \omega_0^2$$

we get:

$$\frac{dx}{dt} = -2\delta x - \frac{\omega_0^2}{2\delta} y$$
$$\frac{dy}{dt} = 2\delta x$$

with the characteristic equation:

$$\lambda^2 + 2\delta\lambda + \omega_0^2 = 0$$

The determinant $\Delta$ is given by:

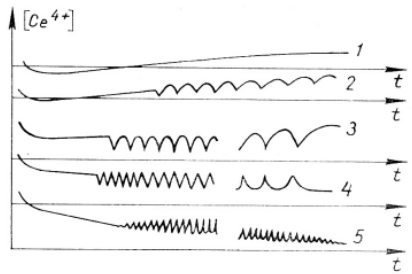
$$\Delta = 4\left(\delta^2 - \omega_0^2\right)$$

Periodic oscillations occur when $\Delta < 0$, i.e. when:

$$\delta^2 < \omega_0^2, \qquad 4k_2^2 > k_0 k_1$$

Experimental confirmation of the occurrence of periodic reactions in nature has been obtained in the 1960s (Zhabotynski *et al.*). Oxidation of malonic acid $C_3H_4O_4$ with a mixture of $KBrO_3$ and $Ce(SO_4)_2$ led to observation of changes in the color of the solution, roughly once per second. It has been shown that the color changes resulted from periodic creation of $Ce^{4+}$ ions in the solution.

The figure below presents some of the results, concerning the observed variations of the $Ce^{4+}$ ions in time. The different curves have been obtained for different initial conditions of the experiment, i.e. for different ratios of concentrations of substrates. The character of the curves varies from sinusoidal (5) to relaxational (2, 3).



## Systèmes proie-prédateur

Lotka (1925) and, independently, Volterra (1926) proposed a simple model for the population dynamics of two interacting species. This model was meant to treat the ***predator-prey interactions***.

Let's assume that hares and lynxes populate an ***isolated area***. Hares eat plants, available without limitations. Lynxes (predators) eat hares (prey). Suppose the numbers of the hares and lynxes are ***X*** and ***Y***, respectively.

With food freely available, hares multiply with the growth rate proportional to their number. However, they perish proportionally to the number of their encounters with lynxes.

Lynxes profit from encounters with the hares, but in the absence of prey, which is their only food, they die out with the rate proportional to their numbers.

Mathematically, we can describe this situation with the help of the following set of equations:

$$\frac{dX}{dt} = rX - aXY$$
$$\frac{dY}{dt} = -mY + bXY$$

where the coefficients ***r***, ***a***, ***m*** and ***b*** are positive constants.

Stationary points of this system can be found if all derivatives are equated to zero:

$$0 = \overline{X}\left(r - a\overline{Y}\right)$$
$$0 = \overline{Y}\left(-m + b\overline{X}\right)$$

The first obvious solution is the stationary point ***(0,0)***. Analysis of small perturbations about this point leads to the following system of equations:

$$\left.\begin{aligned}\frac{dx}{dt} &= rx - axy\\ \frac{dy}{dt} &= -my + bxy\end{aligned}\right\} \xrightarrow{\text{linearization}} \left\{\begin{aligned}\frac{dx}{dt} &= rx\\ \frac{dy}{dt} &= -my\end{aligned}\right.$$

The characteristic equation has the form:

$$\lambda^2 + (m - r)\lambda - rm = 0$$

The solution is:

$$\lambda_1 = -m$$
$$\lambda_2 = r$$

Hence, the solutions [perturbations about the point ***(0,0)***] are linear combinations of the following exponentials:

$$x(t) = Ae^{-mt} + Be^{rt}$$
$$y(t) = Ce^{-mt} + De^{rt}$$

According to the theory, when two roots have different signs the stationary point is a ***saddle***.

The other (non-trivial) solution is:

$$\overline{X} = \frac{m}{b}, \qquad \overline{Y} = \frac{r}{a}$$

Since all parameters are positive, the stationary point is located in the first quadrant of the system's phase plane. Introducing small perturbations ***x(t)*** and ***y(t)*** about this point and linearizing the system gives us the following equations:

$$\frac{dx}{dt} = -\frac{am}{b}y$$
$$\frac{dy}{dt} = \frac{br}{a}x$$

The characteristic equation in this case has the form:

$$\lambda^2 + rm = 0$$

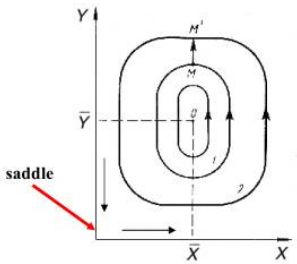
with two imaginary roots:

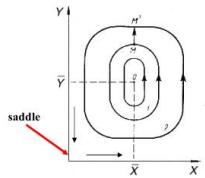
$$\lambda_{1,2} = \pm i\sqrt{rm} = \pm i\omega$$

The solutions are given by the following equations:

$$x(t) = Ae^{i\omega t} + Be^{-i\omega t} = A'\cos(\omega t) + B'\sin(\omega t)$$
$$y(t) = Ce^{i\omega t} + De^{-i\omega t} = C'\cos(\omega t) + D'\sin(\omega t)$$

Imaginary values of the roots indicate that the stationary point is a ***center***. Trajectories close to this point are concentric ellipses. Strictly speaking, however, a center is not a stable point (a center is at the border of a stable and an unstable manifold). Look at the graph below.



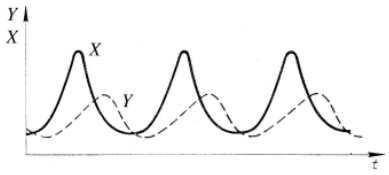




Suppose that the system moves along trajectory ***1***. When it reaches the point ***M***, a certain number of predators are added to the system. As a result, the system's state jumps to the point ***M'***. If no further perturbations occur, oscillations of ***X*** and ***Y*** will continue with larger amplitudes. The new trajectory is now described by curve ***2***.

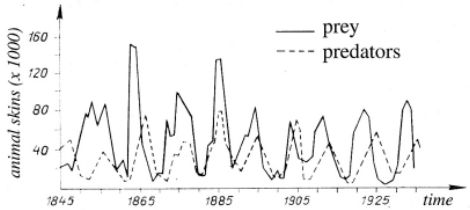
This is a typical case of a ***neutral equilibrium***. Oscillations are unstable because their characteristics change irreversibly under the influence of even small perturbations. A similar type of equilibrium is exhibited by a ball on a flat surface.

**Note**: ***an unstable state cannot describe a real biological system. Hence, despite its simplicity, the Lotka-Volterra model is of limited use.***

In the figure below are shown the temporal changes of ***X*** and ***Y***, obtained by numerical integration. Both functions are periodic. Maximal number of prey ***X*** always precedes the maxima of predators ***Y***.




And here are some experimental data. The Hudson Bay Company clerks have kept logs of animal skins supplied by Canadian trappers beginning with 1845. As can be seen, oscillations occur with periods of about 9 – 10 years. The maximum number of hares usually precedes the maximum number of lynxes by one year. The real-life curves are not as smooth as the simulated ones for obvious reasons: our model does not take into account other predators and/or prey, nor climactic changes.



M2I BBS

# Modélisation et Simulation de Systèmes Biologiques

(#8 - exercices)

Georges Czaplicki, UPS / IPBS-CNRS
Tél. : 05.61.17.54.04, email : cgeorge@ipbs.fr

## Adimensionnement

Conversion de variables d'une équation de façon à se libérer des unités (ce qui est équivalent au changement d'échelle).

**Exemple :**

**Equation :** $A\frac{dx}{dt} + Bx = Cf(t)$

**Substitution :** $x = aX$, $t = bT$

**Par conséquent, les dérivées sont :** $dx = a \cdot dX$, $dt = b \cdot dT$

**Nouvelle forme d'équation :**

$$\left(\frac{Aa}{b}\right)\frac{dX}{dT} + (Ba)X = Cf(bT) \xrightarrow{def} CF(T)$$

**Pour se débarrasser du coefficient du terme dX/dT, on divise tout par (*Aa/b*) :**

$$\frac{dX}{dT} + \left(\frac{bB}{A}\right)X = \left(\frac{bC}{Aa}\right)F(T)$$

**Pour mettre le coefficient du terme X à 1, on choisi :**

$$\frac{bB}{A} = 1 \quad \Rightarrow \quad b = \frac{A}{B}$$

**L'équation devient :** $\frac{dX}{dT} + X = \frac{C}{aB}F(T)$

**Il reste à choisir le coefficient *a* pour simplifier l'expression :**

$$\frac{C}{aB} = 1 \quad \Rightarrow \quad a = \frac{C}{B}$$

**Finalement, l'équation initiale prend la forme suivante :**

$$\frac{dX}{dT} + X = F(T)$$

**où** $X = \frac{B}{C}x$, $T = \frac{B}{A}t$

**La forme simplifiée permet de se focaliser sur la recherche de propriétés de la solution, indépendamment de l'échelle de variables.**

**Sélection d'une des espèces équivalentes**

All living organisms make use of only one of two possible isomers of sugars (***D***, *dextrorotatory*) and amino acids (***L***, *levorotatory*). The other, 'mirror' type of isomer, when present in an organism, may not be assimilated or even become harmful. And yet, those 'mirror' compounds should exhibit similar properties and thus 'live' the same way as their current biological counterparts.

We may argue that at some point in the past both isomers coexisted in a racemic mixture. In the crucial moment of life formation either ***L*** or ***D*** isomers might emerge with equal probability.

We will construct a model describing a process, which results in the survival of one of two equivalent species and the death of the other one. Let ***X*** and ***Y*** be the concentrations of optically active substances, e.g. corresponding to isomers ***L*** and ***D***, respectively. As usual, the growth of each population is proportional to its number, while their antagonistic interaction is modeled by the inclusion of the term ***XY***. The set of equations has the following form:

$$\frac{dX}{dt} = aX - \gamma XY$$
$$\frac{dY}{dt} = aY - \gamma XY$$

The parameter ***a*** balances the reproduction and mortality for each species. The fact that both of the ***XY*** terms are negative means that both objects perish as a result of their antagonistic encounter. The coefficients ***a*** & $\gamma$ are the same for both species, which reflects their equivalence.

To simplify the analysis we will transform the variables to a new dimensionless form, using the following relations:

$$\tau = at, \quad x = \frac{\gamma}{a} X, \quad \mathrm{y} = \frac{\gamma}{\mathrm{a}} Y$$

Here is the new set of equations:

$$\frac{dx}{d\tau} = x - xy$$
$$\frac{dy}{d\tau} = y - xy$$

The stationary points can be found from:

$$0 = x - xy = x(1-y)$$
$$0 = y - xy = y(1-x)$$

There are two of them: ***(0,0***) and ***(1,1)***. Let's take a look at the first one. The theory of perturbations applied to this case gives the following:

$$x(\tau) = 0 + \xi(\tau), \quad \frac{dx}{d\tau} = \frac{d\xi}{d\tau}$$
$$y(\tau) = 0 + \eta(\tau), \quad \frac{dy}{d\tau} = \frac{d\eta}{d\tau}$$

Substituting these expressions into the original set of (dimensionless) equations leads to the following:

$$\frac{d\xi}{d\tau} = \xi - \xi\eta$$
$$\frac{d\eta}{d\tau} = \eta - \xi\eta$$

Linearizing this set of equations gives the following solution:

$$\left.\begin{aligned}\frac{d\xi}{d\tau}&=\xi\\ \frac{d\eta}{d\tau}&=\eta\end{aligned}\right\} \Rightarrow \det\begin{vmatrix}1-\lambda & 0\\ 0 & 1-\lambda\end{vmatrix}=0 \Rightarrow \begin{cases}\lambda_1=1\\ \lambda_2=1\end{cases} \Rightarrow \begin{cases}\xi=Ae^{\tau}\\ \eta=Be^{\tau}\end{cases}$$

Since both roots are positive, the point ***(0,0)*** is an unstable node (all trajectories leave it permanently).

Now let's analyze the second stationary point ***(1,1)***:

$$x(\tau)=1+\xi(\tau),\quad \frac{dx}{d\tau}=\frac{d\xi}{d\tau}$$
$$y(\tau)=1+\eta(\tau),\quad \frac{dy}{d\tau}=\frac{d\eta}{d\tau}$$

Substitution and linearization yields:

$$\frac{d\xi}{d\tau}=-\eta$$
$$\frac{d\eta}{d\tau}=-\xi$$

The solution of the characteristic equation is:

$$\left.\begin{aligned}\frac{d\xi}{d\tau}&=-\eta\\ \frac{d\eta}{d\tau}&=-\xi\end{aligned}\right\} \Rightarrow \det\begin{vmatrix}-\lambda & -1\\ -1 & -\lambda\end{vmatrix}=0 \Rightarrow \begin{cases}\lambda_1=1\\ \lambda_2=-1\end{cases} \Rightarrow \begin{cases}\xi=Ae^{\tau}+Be^{-\tau}\\ \eta=Ce^{\tau}+De^{-\tau}\end{cases}$$

Since the roots have opposite signs, the point ***(1,1)*** is a saddle.

To complete the analysis, let's create a phase portrait of the system. The ***(x,y)*** dependence is obtained by eliminating time from the set of the original equations:

$$\left.\begin{aligned}\frac{dx}{d\tau}&=x-xy\\ \frac{dy}{d\tau}&=y-xy\end{aligned}\right\} \Rightarrow \frac{dy}{dx}=\frac{y(1-x)}{x(1-y)}=\frac{Q(x,y)}{P(x,y)}$$

First, let's find the principal ***isoclines***, i.e. the isoclines of horizontal [***Q(x,y) = 0***] and vertical [***P(x,y) = 0***] adjacent lines:

$$Q(x,y)=0 \Rightarrow y(1-x)=0 \Rightarrow \begin{cases}x=1\\ y=0\end{cases}$$

$$P(x,y)=0 \Rightarrow x(1-y)=0 \Rightarrow \begin{cases}x=0\\ y=1\end{cases}$$

A straight line called ***separator***, which passes through the saddle, separates different attraction basins for the trajectories. Its equation is obtained by assuming that the exponentially growing term is eliminated due to specific initial conditions:

$$\left.\begin{aligned}\xi&=Ae^{\tau}+Be^{-\tau}\\ \eta&=Ce^{\tau}+De^{-\tau}\end{aligned}\right\} \xrightarrow{A=C=0} \frac{\eta}{\xi}=\frac{D}{B} \Rightarrow \eta=\left(\frac{D}{B}\right)\xi$$

As can be seen, the separator passes through the saddle, and also through the origin of the reference frame.

The last element we need to construct the phase portrait of our system is the behaviour of trajectories far from stationary points. Isoclines help us deal with this problem to some extent. However, it is advantageous to know the signs of the ***dy/dx*** derivatives in the phase plane.

$$\frac{dy}{dx} = \frac{y(1-x)}{x(1-y)}$$

In order for the above expression to be positive, the following relationships have to be satisfied:

$$\frac{dy}{dx} > 0 \quad \Rightarrow \quad \frac{y(1-x)}{x(1-y)} > 0 \quad \Rightarrow \quad \begin{Bmatrix} 1-x>0 & \text{AND} & 1-y>0 \\ & \text{OR} & \\ 1-x<0 & \text{AND} & 1-y<0 \end{Bmatrix}$$

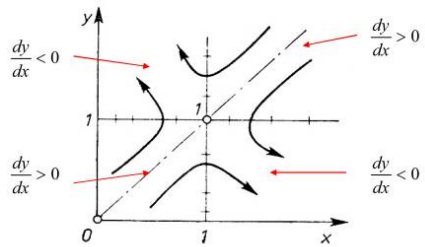
$$\Rightarrow \begin{cases} x<1 & \text{AND} \quad y<1 \\ x>1 & \text{AND} \quad y>1 \end{cases}$$

In order for the derivative to be negative, the following relationships have to be satisfied:

$$\frac{dy}{dx} < 0 \quad \Rightarrow \quad \frac{y(1-x)}{x(1-y)} < 0 \quad \Rightarrow \quad \begin{Bmatrix} 1-x>0 & \text{AND} & 1-y<0 \\ & \text{OR} & \\ 1-x<0 & \text{AND} & 1-y>0 \end{Bmatrix}$$

$$\Rightarrow \begin{cases} x<1 & \text{AND} \quad y>1 \\ x>1 & \text{AND} \quad y<1 \end{cases}$$

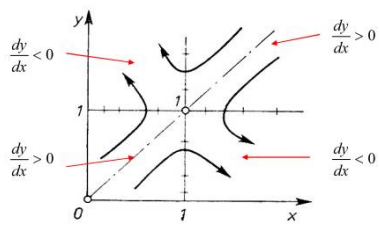
Finally, we are ready to sketch the phase portrait of the system:




It should be easy to see that this system represents a ***switch*** or a ***trigger***. Depending on the initial conditions the trajectories tend to one of the two stable points located in the infinity:

$$\bar{x} = \infty, \qquad \bar{y} = 0$$
$$\bar{x} = 0, \qquad \bar{y} = \infty$$

The interpretation of this result is straightforward: in our model the food resources are unlimited, hence the populations can grow endlessly.




The fact that the stationary point ***(1,1)*** is a saddle is significant: a racemic mixture of species is unstable and arbitrarily small perturbations will cause either an unlimited growth of ***x*** and disappearance of ***y***, or growth of ***y*** and disappearance of ***x***.

***The lack of stability of a symmetric stationary state is the reason for the biologic asymmetry.***

## Exercice : modèle S-I-R

Suppose there is a small group of individuals who are infected with a contagious disease and who have come into a large population. If the population is divided into three groups, the susceptible (S), the infected (I), and the recovered (R), we have what is known as a **classical S-I-R problem**.

The susceptible class consists of those who are not infected, but who are capable of catching the disease and becoming infected. The probability ***r*** of catching a disease upon encountering an infected individual is called the *infection rate*.

The infected class consists of the individuals who are capable of transmitting the disease to others. Their number decreases when they recover. The recovery rate ***a*** is called the *removal rate*.

Finally, the recovered class consists of those who have had the disease, but are no longer infectious.

a) Write the equations and analyze the model (stationary points, stability, trajectories…).
b) Draw graphs for solutions. Use ***r*** = ***a*** = 1. What is the behavior of each class?

**Equations d'après le modèle :**

$$\frac{dS}{dt} = -rSI$$
$$\frac{dI}{dt} = rSI - aI$$
$$\frac{dR}{dt} = aI$$

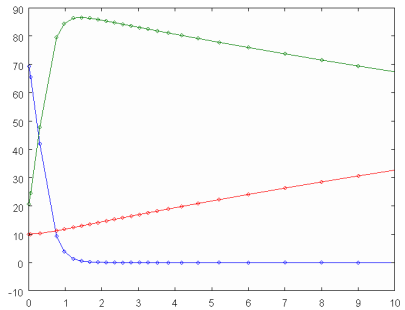
**Fonction Matlab/Freemat qui calcule les trois équations (avec les paramètres *r* & *a*) :**

```
function yp = SIR(t,y,r,a)
yp = zeros(3,1);
yp(1) = -r * y(1) * y(2);
yp(2) =  r * y(1) * y(2) – a * y(2);
yp(3) =  a * y(2);
```

Solution numérique de ce jeu d'équations différentielles :

```
[t,y]=ode45(func,tspan,y0,options,params)
```

```
--> [t,y]=ode45('SIR',[0,10],[70 20 10],['RelTol',1e-6],0.05,0.03);
--> plot(t,y,'-d')
```




**Extrait de commentaires de la commande "ode45" :**

```
% The optional argument 'options' is a structure. It may contain any of the
% following fields:
%
% 'AbsTol' - Absolute tolerance, default is 1e-6.
% 'RelTol' - Relative tolerance, default is 1e-3.
% 'MaxStep' - Maximum step size, default is (tspan(2)-tspan(1))/10
% 'InitialStep' - Initial step size, default is maxstep/100
% 'Stepper' - To override the default Fehlberg integrator
% 'Events' - To provide an event function
% 'Projection' - To provide a projection function
```

## Modèle S-I-R avec perte d'immunité

**Equations modifiées :**

$$\frac{dS}{dt} = -rSI + bR(t-\tau)$$

$$\frac{dI}{dt} = rSI - aI$$

$$\frac{dR}{dt} = aI - bR(t-\tau)$$

**Paramètres :**
**r = taux d'infection (probabilité d'être infecté lors d'un contact avec I)**
**a = taux de récupération (l'inverse de temps de guérison)**
**b = pourcentage de récupérés qui perdent leur immunité après le délai $\tau$.**

## Solving Delay Differential Equations with `dde23`

Ordinary differential equations (ODEs) and delay differential equations (DDEs) are used to describe many phenomena of physical interest. While ODEs contain derivatives which depend on the solution at the present value of the independent variable ("time"), DDEs contain in addition derivatives which depend on the solution at previous times. DDEs arise in models throughout the sciences [1]. Despite the obvious similarities between ODEs and DDEs, solutions of DDE problems can differ from solutions for ODE problems in several striking, and significant, ways [2] [20]. This accounts in part for the lack of much general-purpose software for solving DDEs.

$$y'(t) = f(t, y(t), y(t-\tau_1), y(t-\tau_2), \ldots, y(t-\tau_k))$$

## Example 1

We illustrate the straightforward solution of a DDE by computing and plotting the solution of Example 3 of [23]. The equations

$$\begin{aligned} y_1'(t) &= y_1(t-1) \\ y_2'(t) &= y_1(t-1) + y_2(t-0.2) \\ y_3'(t) &= y_2(t) \end{aligned}$$

are to be solved on $[0, 5]$ with history $y_1(t) = 1, y_2(t) = 1, y_3(t) = 1$ for $t \leq 0$.

A typical invocation of `dde23` has the form

```
sol = dde23(ddefile,lags,history,tspan);
```

**MATLAB !**

The input argument `tspan` is the interval of integration, here `[0, 5]`. The `history` argument is the name of a function that evaluates the solution at the input value of `t` and returns it as a column vector. Here `exam1h.m` can be coded as

```
function v = exam1h(t)
v = ones(3,1);
```

**OCTAVE**

**1.** Installer **odepkg** dans Octave :

```
> pkg install -forge odepkg
> pkg load odepkg
```

**2.** Si le package n'est pas trouvé sur le site SourceForge :

**- télécharger *odepkg-0.8.5.tar.gz* de SourceForge**
**- cd vers le répertoire où se trouve l'archive *odepkg***
**- `pkg install odepkg-0.8.5.tar.gz`**
**- `pkg load odepkg`**

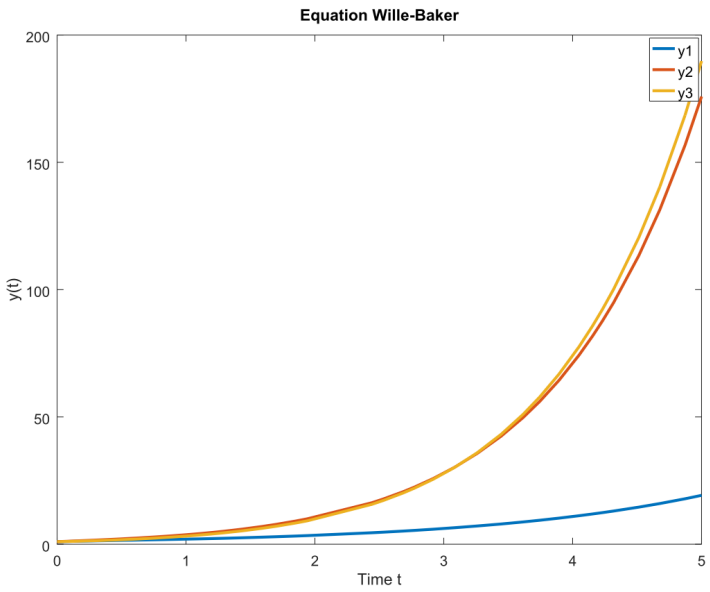
**3.** Ceci va rendre disponibles les routines pour ODE et DDE.

**Exemple OCTAVE**

```
function dydt=solveWB(t,y,Z)
  ylag1=Z(:,1);
  ylag2=Z(:,2);
  dydt=[ylag1(1); ylag1(1)+ylag2(2); y(2)];
```

```
sol=ode54d(@solveBW,[0 5],[1 1 1],[1,0.2],ones(3,2));

%Options : ^fonction
%                    ^temps de simulation
%                           ^valeurs initiales
%                                   ^lags
%                                         ^historique

plot(sol.x, sol.y, 'linewidth',2)
legend('y1','y2','y3')
title('Equation Wille-Baker')
xlabel('Time t')
ylabel('y(t)')
```

**Exemple OCTAVE**



**Exemple OCTAVE : SIR avec DDE**

Equations d'après le modèle :

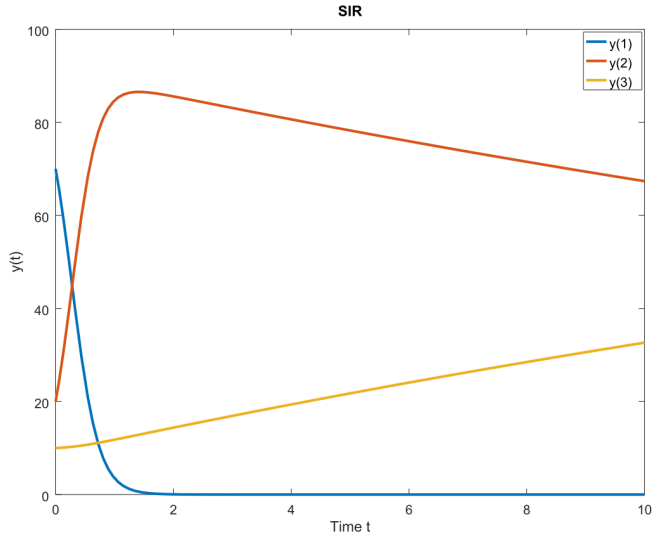
$$\frac{dS}{dt} = -rSI$$
$$\frac{dI}{dt} = rSI - aI$$
$$\frac{dR}{dt} = aI$$

Fonction Octave
qui calcule les trois équations
(avec les paramètres *r* & *a*) :

```
function yp = SIR(t,y,Z,r,a)
yp = zeros(3,1);
yp(1) = -r * y(1) * y(2);
yp(2) =  r * y(1) * y(2) – a * y(2);
yp(3) =  a * y(2);
```

```
opt=odeset()
opt =
  scalar structure containing the fields:
    RelTol = [](0x0)
    AbsTol = [](0x0)
    NormControl = off
    NonNegative = [](0x0)
    OutputFcn = [](0x0)
    OutputSel = [](0x0)
    OutputSave = [](0x0)
    Refine = 0
    Stats = off
    InitialStep = [](0x0)
    MaxStep = [](0x0)
    Events = [](0x0)
    Jacobian = [](0x0)
    JPattern = [](0x0)
    Vectorized = off
    Mass = [](0x0)
    MStateDependence = weak
    MvPattern = [](0x0)
    MassSingular = maybe
    InitialSlope = [](0x0)
    MaxOrder = [](0x0)
    BDF = [](0x0)
    NewtonTol = [](0x0)
    MaxNewtonIterations = [](0x0)
```

```
r=0.05; a=0,03;

opt=odeset('RelTol',1e-6,'AbsTol',1e-9,'InitialStep',0.5,'MaxStep',0.5);

sol=ode54d(@sir,[0 10],[70 20 10],[1],ones(3,1),opt,r,a);
```




## Problem 1

Hale [7] cites predator–prey models obtained by introducing a resource limitation on the prey and assuming the birth rate of predators responds to changes in the magnitude of the population $y_1$ of prey and the population $y_2$ of predators only after a time delay $\tau$. Starting with the system of ODEs [15]

$$\begin{aligned} y_1'(t) &= a\,y_1(t) + b\,y_1(t)\,y_2(t) \\ y_2'(t) &= c\,y_2(t) + d\,y_1(t)\,y_2(t) \end{aligned}$$

we arrive in this way at a system of DDEs

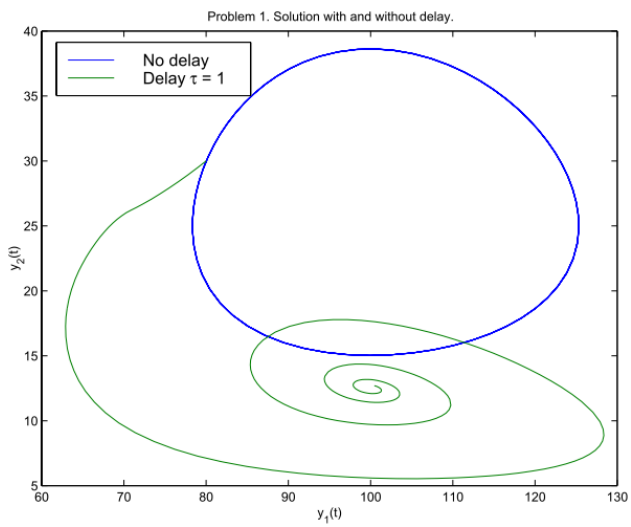
$$\begin{aligned} y_1'(t) &= a\,y_1(t)\left(1 - \frac{y_1(t)}{m}\right) + b\,y_1(t)\,y_2(t) \\ y_2'(t) &= c\,y_2(t) + d\,y_1(t-\tau)\,y_2(t-\tau) \end{aligned}$$

It is interesting to explore the effect of the delay, so let us solve both systems on $[0, 100]$ with initial value $y_1(0) = 80, y_2(0) = 30$ for the ODEs and the same vector as constant history for the DDEs. Suppose that the parameters $a = 0.25, b = -0.01, c = -1.00, d = 0.01$, and $m = 200$.

You will need to tighten the error tolerances with a command like

```
opts = ddeset('RelTol',1e-5,'AbsTol',1e-8);
```

The figure makes the point that introducing a delay into an ODE model can have a profound effect on the solution. If you experiment with $\tau$, you will find this to be true even for small delays. It is also interesting to remove the resource term $1 - y_1(t)/m$ and see how the orbits change as $\tau$ is changed.



## Modèle S-I-R avec perte d'immunité

**Equations modifiées :**

$$\frac{dS}{dt} = -rSI + bR(t-\tau)$$

$$\frac{dI}{dt} = rSI - aI$$

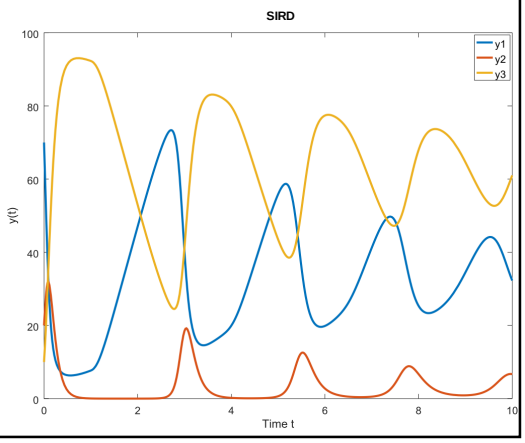
$$\frac{dR}{dt} = aI - bR(t-\tau)$$

**Paramètres :**
**r = taux d'infection (probabilité d'être infecté lors d'un contact avec I)**
**a = taux de récupération (l'inverse de temps de guérison)**
**b = pourcentage de récupérés qui perdent leur immunité après le délai $\tau$.**

## Exemple OCTAVE

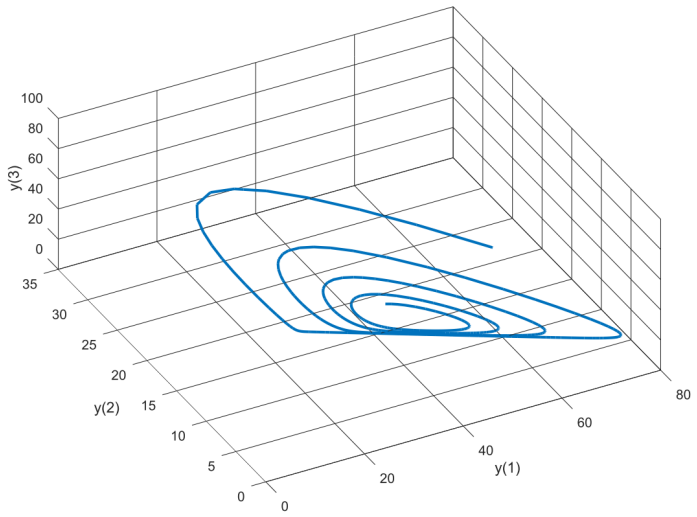
```
r=0.3; a=10; b=0.5;

opt=odeset('RelTol',1e-6,'AbsTol',1e-9,'InitialStep',0.5,'MaxStep',0.5);

sol=ode54d(@sird,[0 10],[70 20 10],[1],ones(3,1),opt,r,a,b);
```

```
function dy = sird(t,y,Z,r,a,b)
dy=zeros(3,1);
ylag=Z(:,1);
dy(1)=-r*y(1)*y(2)+b*ylag(3);
dy(2)= r*y(1)*y(2)-a*y(2);
dy(3)= a*y(2)-b*ylag(3);
```



## Exemple OCTAVE : image de phase (plot 3D)

```
plot3(sol.y(:,1),sol.y(:,2),sol.y(:,3),'linewidth',2)
grid on
xlabel('y(1)')
ylabel('y(2)')
zlabel('y(3)')
```




## Modèle S-I-R avec perte d'immunité

Equations modifiées (fonction `sird`) :

$$\frac{dS}{dt} = -rSI + bR(t-\tau)$$

$$\frac{dI}{dt} = rSI - aI$$

$$\frac{dR}{dt} = aI - bR(t-\tau)$$

**Résoudre les équations utilisant les jeux de paramètres suivants :**

**1) $\tau = 1$; r = 0.5; a = 20; b=1**
**2) $\tau = 1$; r = 0.5; a = 20; b=0,5**
**3) $\tau = 1$; r = 0.3; a = 20; b=0,5**

**Explorer d'autres jeux de paramètres.**

**Visualisez les résultats. Conclusions ?**

## Modèle S-I-R avec perte d'immunité et avec incubation

**Equations modifiées (fonction `SIRD2`) :**

$$\frac{dS}{dt} = -rSI(t-\tau_1) + bR(t-\tau_2)$$

$$\frac{dI}{dt} = rSI(t-\tau_1) - aI$$

$$\frac{dR}{dt} = aI - bR(t-\tau_2)$$

**Ecrire les équations et les résoudre avec dde23, utilisant les jeux de paramètres suivants :**

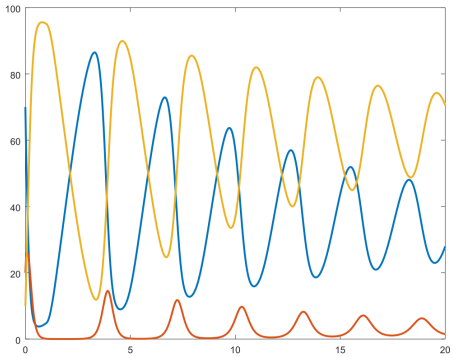
**1) $\tau_1$ = 0.1; $\tau_2$ = 1; r = 0.3; a = 10; b=0,5**
**2) $\tau_1$ = 1; $\tau_2$ = 1; r = 0.3; a = 10; b=0,5**
**3) $\tau_1$ = 2; $\tau_2$ = 1; r = 0.3; a = 10; b=0,5**

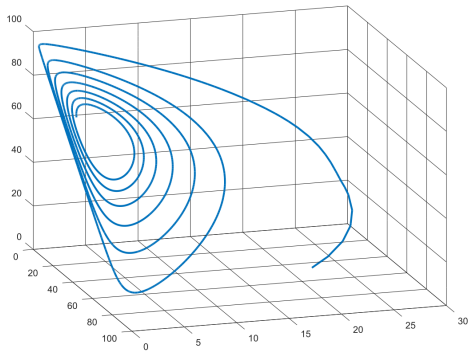
**Visualisez les résultats. Conclusions ?**

**Solution de l'exemple SIRD2**
**(à voir APRES avoir essayé de le résoudre soi-même !)**

```
function dy = sird2(t,y,Z,r,a,b)
dy=zeros(3,1);
ylag1=Z(:,1);
ylag2=Z(:,2);
dy(1)=-r*y(1)*ylag1(2)+b*ylag2(3);
dy(2)= r*y(1)*ylag1(2)-a*y(2);
dy(3)= a*y(2)-b*ylag2(3);
```

```
r=0.3; a=10; b=0.5;
sol=ode54d(@sird2,[0 20],[70 20 10],[0.1 1],ones(3,2),opt,r,a,b);
```



## Modèle prédateur-proie avec incubation

$$\frac{dS}{dt} = rS\left(1 - \frac{S}{K}\right) - bSI(t-\tau)$$

$$\frac{dI}{dt} = bSI(t-\tau) - \frac{cPI}{mP + I}$$

$$\frac{dP}{dt} = dP(1 - h\frac{P}{I})$$

**Proie : susceptible (S) ou infectée (I), croissance logistique.**

**Prédateur : attrape plutôt les proies malades, croissance logistique.**

**Ecrire les équations et les résoudre avec dde23, utilisant le jeux de paramètres suivant :**

**r = 0.1; K=500; b=0.001; c=8; m=150; d=0.2; h=0.5**

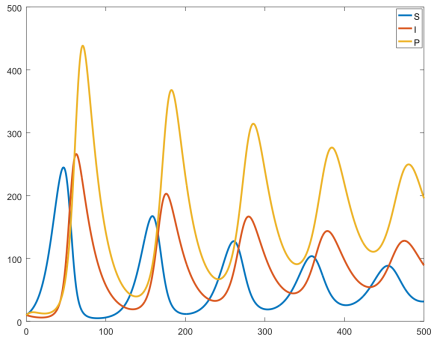
**pour les valeurs de $\tau$ suivantes: $\tau$ = 2, $\tau$ = 10, $\tau$ = 100, $\tau$ = 200.**

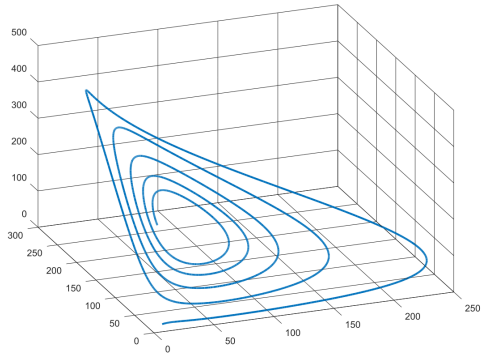
**Visualisez les résultats. Conclusions ?**

**Solution de PPID**
**(à voir APRES avoir essayé de le résoudre soi-même !)**

```
function dy=ppid(t,y,Z,p)
  dy=zeros(3,1);
  ylag=Z(:,1);
  r=p(1); K=p(2); b=p(3); c=p(4);
  m=p(5); d=p(6); h=p(7);
  dy(1)=r*y(1)*(1-y(1)/K)-b*y(1)*ylag(2);
  dy(2)=b*y(1)*ylag(2)-c*y(3)*y(2)/(m*y(3)+y(2));
  dy(3)=d*y(3)*(1-h*y(3)/y(2));
```

```
p=[0.1 500 0.001 8 150 0.2 0.5];
sol=ode54d(@ppid,[0 500],[10 10 10],[1],ones(3,1),opt,p);
plot(sol.x,sol.y,'linewidth',2); legend('S','I','P')
plot3(sol.y(:,1),sol.y(:,2),sol.y(:,3),'linewidth',2); grid on
```



Exercice

Montrer que la linéarisation du système :

$$\frac{dx}{dt} = -y + ax \cdot \left(x^2 + y^2\right)$$
$$\frac{dy}{dt} = x + ay \cdot \left(x^2 + y^2\right)$$

prédit que le seul point stationnaire est un centre pour toutes les valeurs du paramètre *a*, mais en réalité c'est une spirale stable si *a* < 0 et une spirale instable si *a* > 0. Tracez les portraits de phase de ce système.

Le point stationnaire peut être obtenu des équations suivantes :

$$0 = -y + ax \cdot \left(x^2 + y^2\right)$$
$$0 = x + ay \cdot \left(x^2 + y^2\right)$$

On regroupant les termes on arrive à :

$$\frac{y}{ax} = \left(x^2 + y^2\right)$$
$$\frac{-x}{ay} = \left(x^2 + y^2\right)$$

En comparant les termes à gauche du "=" on a :

$$\boxed{\frac{y}{ax} = \frac{-x}{ay}}$$ d'où on obtient : $ay^2 = -ax^2$

et finalement : $\boxed{a \cdot \left(x^2 + y^2\right) = 0}$ donc : $\boxed{(0,0)}$

Linéarisation du système :

$$\left.\begin{array}{l} x = 0 + \xi \\ y = 0 + \eta \end{array}\right\} \qquad \begin{array}{l} \dfrac{d\xi}{dt} = -\eta + a\xi\left(\xi^2 + \eta^2\right) \\ \dfrac{d\eta}{dt} = \xi + a\eta\left(\xi^2 + \eta^2\right) \end{array}$$

Si on ne retient que les termes linéaires :

$$\boxed{\begin{array}{l} \dfrac{d\xi}{dt} = -\eta \\ \dfrac{d\eta}{dt} = \xi \end{array}}$$

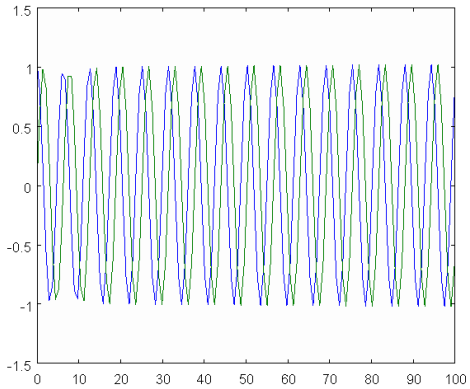
Pour trouver les racines du polynôme caractéristique :

$$A = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \qquad \det(A - \lambda I) = \begin{bmatrix} -\lambda & -1 \\ 1 & -\lambda \end{bmatrix} = 0 \qquad \left\{\begin{array}{l} \lambda^2 + 1 = 0 \\ \lambda_{1,2} = \pm i \end{array}\right.$$

$$\boxed{\begin{pmatrix} \xi \\ \eta \end{pmatrix} = C_1 e^{it} + C_2 e^{-it} = C_1 \cos(t) + C_2 \sin(t)}$$

```
function yp = myfcn(t,y,a)
yp=zeros(2,1);
yp(1) = -y(2) + a*y(1)*(y(1)^2+y(2)^2);
yp(2) =  y(1) + a*y(2)*(y(1)^2+y(2)^2);
```

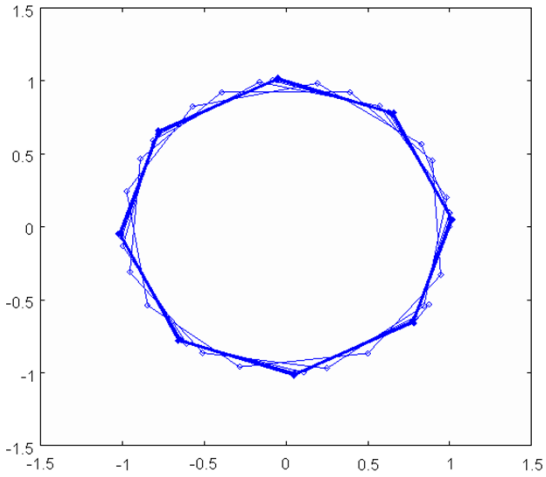
```
--> [t,y]=ode45('myfcn',[0,100],[1,0],[],0);
--> plot(t,y)
```



```
--> y
ans =
    1.0000         0
    0.9950    0.0998
    0.8253    0.5646
    0.1919    0.9816
   -0.5697    0.8223
   -0.9713    0.2400
   -0.8430   -0.5391
   -0.2816   -0.9604
    0.5006   -0.8668
    0.9448   -0.3310
    0.8933    0.4524
    0.3905    0.9221
   -0.3913    0.9220
   -0.8881    0.4632
   -0.9521   -0.3117
   -0.5075   -0.8640
    0.2497   -0.9705
    0.8424   -0.5431
    0.9822    0.2001
    0.5702    0.8247
   -0.1614    0.9896
   -0.8105    0.5907
   -0.9943   -0.1318
   -0.6060   -0.7995
    0.1094   -0.9974
    0.7912   -0.6174
    0.9994    0.0927
    0.6257    0.7850
   -0.0804    1.0008
   -0.7805    0.6319
```

Pour obtenir l'image de phase du système :

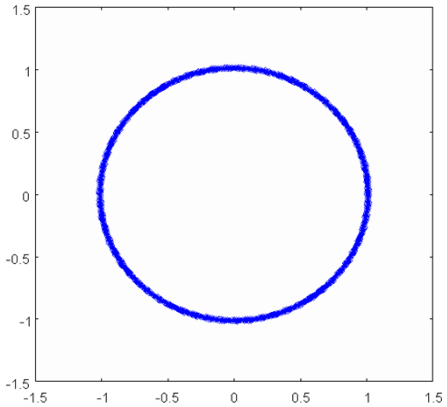
```
--> plot(y(:,1),y(:,2),'-d')
```




Pour avoir plus de points pour le graphe :

```
--> SOL=ode45('myfcn',[0,100],[1,0],[],0);
--> y=deval(SOL,0:0.1:100);
--> y=y';
--> plot(y(:,1),y(:,2),'-d')
```

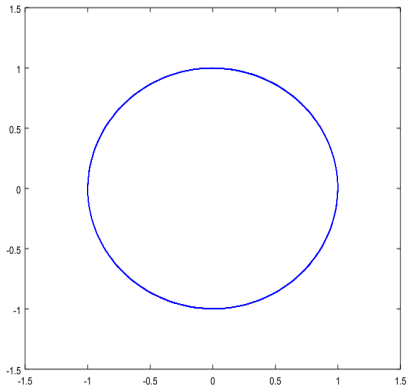
**MATLAB**



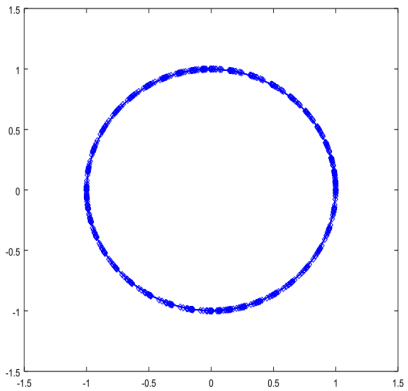
**OCTAVE**

```
opt=odeset('RelTol',1e-6,'AbsTol',1e-9,'InitialStep',0.5,'MaxStep',0.5);
[t,y]=ode45(@myfcn,[0,100],[1,0],opt,0);
```

```
plot(y(:,1),y(:,2))
```



```
plot(y(:,1),y(:,2),'-d');
```



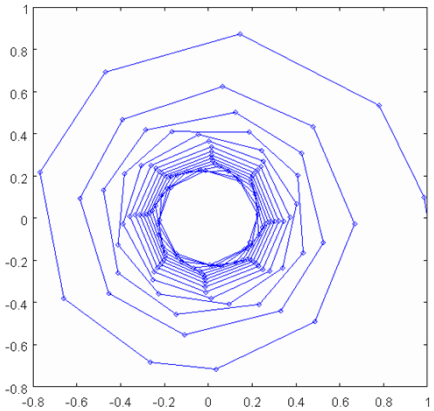
**Autre possibilité :**

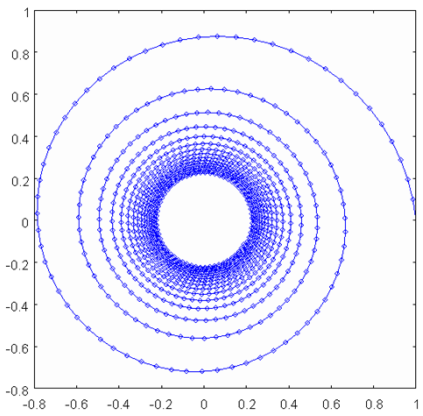
```
sol=ode45(@myfcn,[0,100],[1,0],opt,0);
plot(sol.y(:,1),sol.y(:,2))
```

**MATLAB**

Pour voir ce qui se passe quand a < 0 :

```
--> [t,y]=ode45('myfcn',[0,100],[1,0],[],-0.1);
--> plot(y(:,1),y(:,2),'-d')
```



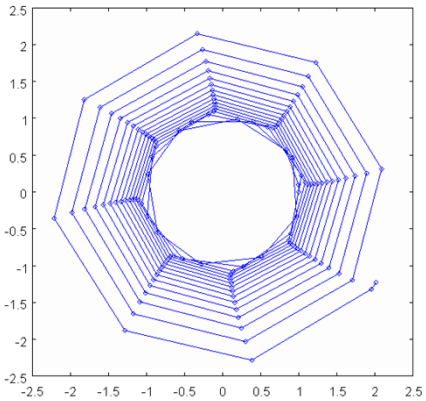


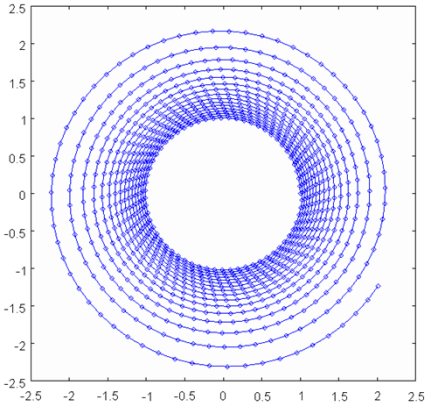
(SOL)

**MATLAB**

Pour voir ce qui se passe quand a > 0 :

```
--> [t,y]=ode45('myfcn',[0,100],[1,0],[],0.004);
--> plot(y(:,1),y(:,2),'-d')
```

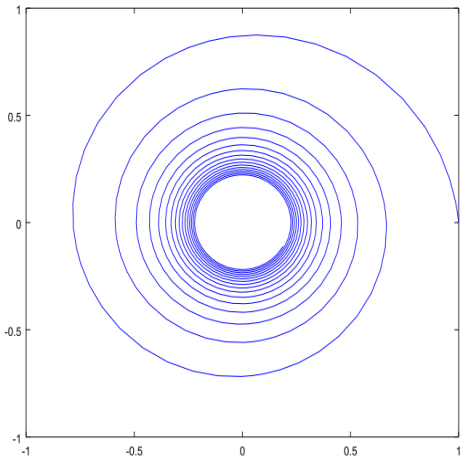



(SOL)

**OCTAVE**

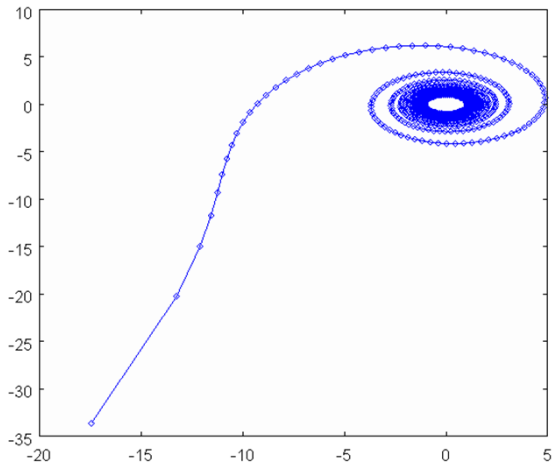
```
a=-0.1;
sol=ode45(@myfcn,[0,100],[1,0],opt,a);
plot(sol.y(:,1),sol.y(:,2))
```



**MATLAB**

**A cause d'une forte non-linéarité du système, l'intégration numérique peut causer quelques problèmes. Exemple : *a* = 0.005 :**

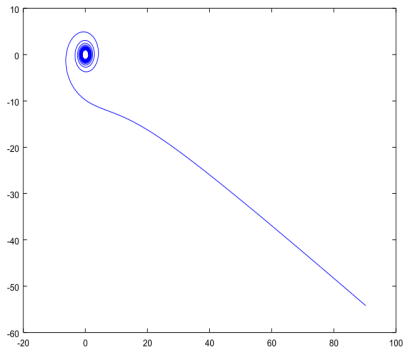
```
--> y
ans =
   1.0e+003 *
    0.0010         0
    0.0010    0.0001
    0.0008    0.0006
    0.0002    0.0010
   -0.0006    0.0008
   -0.0019   -0.0035
    0.0013   -0.0041
    0.0041   -0.0021
    0.0047    0.0017
...
   -0.0208   -0.0428
   -0.0352   -0.0782
   -0.0962   -0.2211
   -0.9499   -2.1944
 -Inf -Inf
 NaN NaN
 NaN NaN
 NaN NaN
 NaN NaN
 NaN NaN
```




(SOL)

**OCTAVE**

```
a=0,005;
sol=ode45(@myfcn,[0,100],[1,0],opt,a);

error: Solving has not been successful. The iterative integration loop
exited at time t = 99.999026 before end point at tend = 100.000000 was
reached. This may happen if the stepsize grows smaller than defined in
vminstep size. Try to reduce the value of "InitialStep" and/or
"MaxStep" with the command "odeset".

>> sol=ode45(@myfcn,[0,99.99],[1,0],opt,a);
>> plot(sol.y(:,1),sol.y(:,2))
```



**Exercice : compétition**

The following *competition model* is provided in Reference [9]. Imagine rabbits and sheep competing for the same limited amount of grass. Assume a logistic growth for the two populations, that rabbits reproduce rapidly, and that the sheep will crowd out the rabbits. Assume these conflicts occur at a rate proportional to the size of each population. Further, assume that the conflicts reduce the growth rate for each species, but make the effect more severe for the rabbits by increasing the coefficient for that term. A model that incorporates these assumptions is

$$\frac{dx}{dt} = x(3 - x - 2y)$$

$$\frac{dy}{dt} = y(2 - x - y)$$

where $x(t)$ is the rabbit population and $y$ is the sheep population. (Of course, the coefficients are not realistic, but are chosen to illustrate the possibilities.)

**Exercice : prédation**

$$\frac{dx}{dt} = x(a - bx - cy)$$

$$\frac{dy}{dt} = y(-k + \lambda x - \sigma y)$$

**Proies (x): croissance logistique.**

**Prédateurs (y) : compétition.**

**Trouver les points stationnaires dans le cas général (c-à-d travailler avec les coefficients dans la forme symbolique, sans leur donner les valeurs concrètes). Tous les coefficients sont positifs.**

**Dessiner la solution dans le plan phasique, avec les isoclines et un jeu de Trajectoires orientées.**

**Exercice : exclusion compétitive (1)**

## 3.3 Competitive exclusion

We consider an ordinary differential equation model of two competitors. An example might be populations of red squirrels and grey squirrels [8]. Here, both populations compete for the same resources and a typical model for their dynamics is

$$\frac{dN_1}{dt} = r_1 N_1 \left(1 - \frac{N_1}{K_1} - b_{12}\frac{N_2}{K_1}\right), \tag{3.29}$$

$$\frac{dN_2}{dt} = r_2 N_2 \left(1 - \frac{N_2}{K_2} - b_{21}\frac{N_1}{K_2}\right), \tag{3.30}$$

where $K_1$, $K_2$, $r_1$, $r_2$, $b_{12}$, $b_{21}$ are positive constants. Let us associate $N_1$ with red squirrels and $N_2$ with grey squirrels in our example.

In particular, given a range of parameter values and some initial values for $N_1$ and $N_2$ at the time $t = 0$, we would typically like to know if the final outcome is one of the following possibilities:

- the reds become extinct, leaving the greys;
- the greys become extinct, leaving the reds;
- both reds and greys become extinct;
- the reds and greys co-exist. If this system is perturbed in any way will the reds and greys continue to coexist?

**Exercice : exclusion compétitive (2)**

**Tout d'abord, adimmensionner les équations !**

**Ensuite, analyser le système (points stationnaires, stabilité, isoclines, etc.).**

**Puis, refaire l'analyse du système modifié de façon suivante :**

## 3.4 Mutualism (symbiosis)

We consider the same ordinary differential equation model for two competitors, *i.e.*

$$\frac{dN_1}{dt} = r_1 N_1 \left(1 - \frac{N_1}{K_1} + b_{12}\frac{N_2}{K_1}\right), \tag{3.40}$$

$$\frac{dN_2}{dt} = r_2 N_2 \left(1 - \frac{N_2}{K_2} + b_{21}\frac{N_1}{K_2}\right), \tag{3.41}$$

M2I BBS

# Modélisation et Simulation de Systèmes Biologiques

(#9 – trig. gén.)

Georges Czaplicki, UPS / IPBS-CNRS
Tél. : 05.61.17.54.04, email : cgeorge@ipbs.fr

## Bistable systems

For the network studied in the previous section (model (4.1)), we saw that all trajectories converge to a unique steady state. To explore an alternative asymptotic behaviour, we next consider the network in Figure 4.6. This reaction scheme is symmetric—each species allosterically inhibits production of the other, resulting in a mutual antagonism.

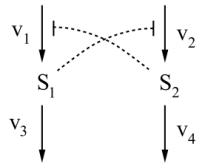



Figure 4.6: Symmetric biochemical network. Each species allosterically inhibits production of the other.

$$\frac{d}{dt}s_1(t) = \frac{k_1}{1+(s_2(t)/K_2)^{n_1}} - k_3 s_1(t)$$
$$\frac{d}{dt}s_2(t) = \frac{k_2}{1+(s_1(t)/K_1)^{n_2}} - k_4 s_2(t)$$

We first consider an asymmetric model parametrization in which $n_1 > n_2$. In this case, the inhibition by $S_2$ is more effective than the inhibition by $S_1$. If the other parameters are symmetric ($k_1 = k_2$, $K_1 = K_2$, $k_3 = k_4$) we should expect the model to exhibit a steady state in which the concentration of $S_1$ is low and the concentration of $S_2$ is high (the mutual antagonism 'competition' will be won by $S_2$). This intuition is confirmed by Figure 4.7. Panel A shows two time-courses starting from different initial conditions. Regardless of whether $S_1$ or $S_2$ is initially more abundant, the imbalance in inhibition strength leads to the same steady state (low $[S_1]$, high $[S_2]$). The phase portrait in Figure 4.7B confirms this finding. All trajectories converge to the steady state at the intersection of the $s_1$- and $s_2$-nullclines, at which $S_2$ dominates.

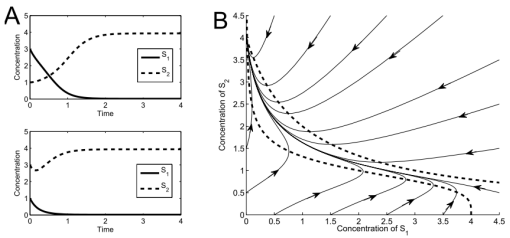



Figure 4.7: Model (4.2) with imbalanced inhibition strength. **A.** Time-series plots show that regardless of the initial condition, the system settles to a steady state with high $S_2$ concentration and low $S_1$ concentration. **B.** This phase portrait confirms that all trajectories approach the high-$[S_2]$, low-$[S_1]$ steady state at which the nullclines (dashed lines) intersect. Parameter values: $k_1 = k_2 = 20$ (concentration$\cdot$time$^{-1}$), $K_1 = K_2 = 1$ (concentration), $k_3 = k_4 = 5$ (time$^{-1}$), $n_1 = 4$, and $n_2 = 1$. Units are arbitrary.

**Symmetric case:** The system's steady-state behaviours are illustrated in Figure 4.8. Panel A shows that the long-time behaviour depends on the initial conditions—whichever species is initially more abundant maintains its dominance. The phase portrait in Panel B shows a symmetric phase plane. Trajectories are attracted to whichever steady state is closer. The region of the phase plane from which trajectories converge to each steady state is called the *basin of attraction* of that steady state. Curves that separate basins of attraction are called *separatrices*. In this perfectly symmetric case, the separatrix is the diagonal.

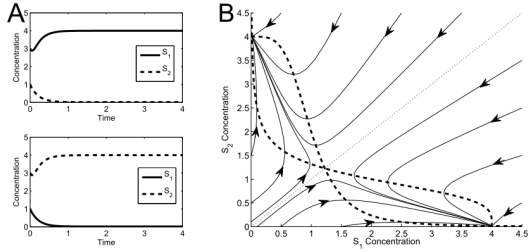



Figure 4.8: Model (4.2) with balanced inhibition strength. **A.** Time-series plots show that the steady state behaviour depends on the initial conditions. Either species can dominate over the other if its initial concentration is larger. **B.** The phase portrait confirms the presence of two steady states at which the nullclines (dashed lines) intersect. Each trajectory converges to the closer steady state. The two basins of attraction are separated by the diagonal (dotted line). Parameter values: $k_1 = k_2 = 20$ (concentration $\cdot$ time$^{-1}$), $K_1 = K_2 = 1$ (concentration), $k_3 = k_4 = 5$ (time$^{-1}$), $n_1 = n_2 = 4$. Units are arbitrary.

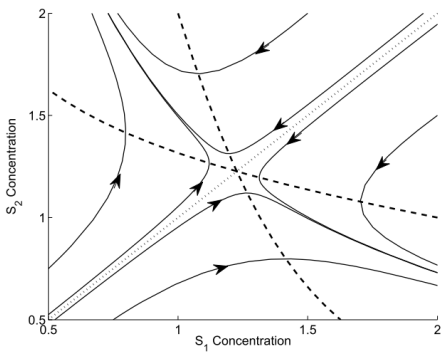



Figure 4.9: Model (4.2) with balanced inhibition strength: unstable steady state. Figure 4.8B shows the nullclines intersecting three times. This close-up of the middle intersection shows that trajectories are repelled from this unstable steady state. The dashed lines are the nullclines. The dotted line (on the diagonal) is the separatrix that forms the boundary between the two basins of attraction.

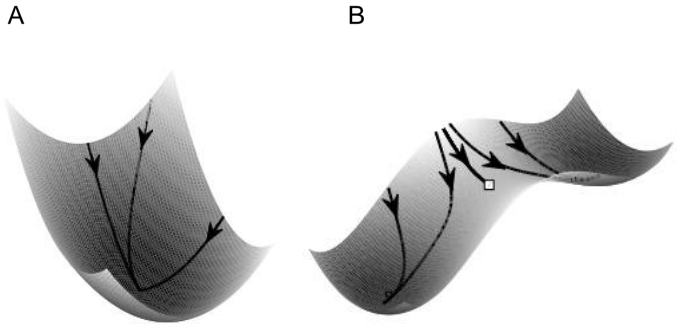



Figure 4.11: Monostability and bistability. **A.** For a monostable system, there is a single valley bottom, representing a unique steady state to which all trajectories converge. **B.** A bistable system corresponds to a pair of valleys, separated by a ridge. The low point of the ridge is the unstable steady state (a saddle point). Most trajectories settle to one of the valley bottoms, but trajectories that remain perfectly balanced on the ridge will settle to the unstable saddle point.

**Triggers (déclencheurs) génétiques**

All cells of a biological organism contain the same genetic information. It is only natural, as they are all formed by divisions of the same initial cell of fertilized egg.

On the other hand, cells of various organs differ both in features and functions. Hence we conclude that functioning of different cells of an adult organism is based on different fragments of the genetic code.

A fertilized cell divides into $10^3 \div 10^4$ new equivalent cells. Then the first ***differentiation*** takes place: cells form groups working in different regimes. At the same time the embryo changes its shape. Further differentiation occurs, leading to a formation of a fully developed organism.

The mechanism, regulating this important process is not yet known. However, we suspect ***cellular self-organization*** to be responsible for this effect. We may find out more by analyzing an appropriate mathematical model.

It seems that an adult cell, working in a specified regime, may occasionally ***switch*** to another regime. Think of cancerous tissues: cells grow rapidly, in a way characteristic of early development.

So far, we have been unable to find ways to control switching cellular regimes. We can, however, artificially affect these processes by introducing certain external factors.

Similar problems arise in the analysis of the ***evolution of biosphere***. Elementary act of evolution consists in creating two different individuals coming from the same ancestor. Analogous to the cellular differentiation, this process should be described by a similar mechanism.

Let's construct and analyze a model, which may give us some insight into the mechanism of differentiation. It is easy to see that our model should allow the existence of at least two stationary points, which would correspond to two different regimes of work under the same external conditions. In other words, the system's phase portrait should contain a certain number of singular points, located at the intersections of isoclines. Such systems are well known in electronics; they are called ***triggering (switching) systems***.

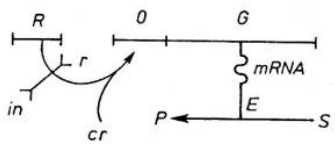
A perfect example of a ***trigger*** is ***protein biosynthesis***. We will base the analysis of this mechanism on the model of its biochemical control given by Jacob and Monod (1961).

Properties of a protein depend on the sequence of amino acids, forming its ***primary structure***. This sequence is determined by the sequence of nucleotides in that part of DNA, which controls the biosynthesis of that particular protein (***structural gene***, ***G***).

Before the structural gene in the DNA sequence, there is the ***operator DNA*** (***O***). The biosynthesis is initiated when RNA polymerase binds to the operator and starts the ***transcription*** by moving along the nucleotide chain and catalyzing the creation of complementary messenger RNA (***mRNA***) molecules.

Next, the ***translation*** takes place, i.e. the synthesis of the protein's amino acid sequence, coded in mRNA, on appropriate ribosomal units. In this process, catalyzed by enzyme ***E***, substrates ***S*** are processed into products ***P***.

The process thus far described can be visualized in the rhs of the figure below.



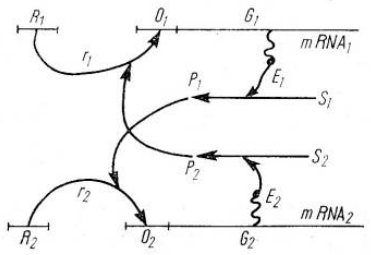

The biosynthesis of a protein proceeds only if the RNA polymerase binds to the operator DNA (***O***). If ***O*** is occupied by another molecule, the biosynthesis is effectively blocked. Enzymes that bind to ***O*** and thus control the biosynthesis are called ***repressors*** (***r***). They are synthesized by another part of DNA, the ***regulator DNA*** (***R***).

Binding of repressors to operator DNA is controlled by two types of other molecules (usually small and not necessarily proteins). The molecules of the first type (***inductors***, ***in***) induce biosynthesis by binding to repressors and thereby preventing them from binding to ***O***. The molecules of the second type (***co-repressors***, ***cr***) inhibit biosynthesis by binding to repressors and thus increasing their affinity for binding to the operator DNA ***O***.

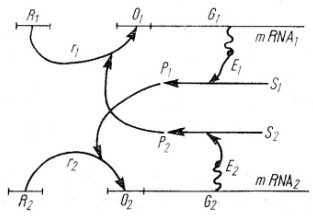
Inductors and co-repressors regulate the process of biosynthesis by providing it with appropriate ***feedback*** mechanisms. In both cases, the actual interaction between these molecules and that of a repressor results in a modification of the structure and/or function of the latter.

In many cases it is the product ***P***, resulting from the activity of enzyme ***E***, which plays the role of a co-repressor. In such a case the synthesis of enzyme ***E*** is interrupted when the concentration of its product ***P*** is high and the enzyme is no longer needed by the cell.

In other cases, a co-repressor may be a product of activity of another enzyme. Jacob and Monod suggested a schema of a system, which plays the role of a ***switch***. It is shown below:




Arrows linking $P_1$ and $r_2$ as well as $P_2$ and $r_1$ indicate that products of activities of enzymes $E_1$ and $E_2$ are each a co-repressor for the synthesis of the 'opposite' enzyme. This system may work in two mutually exclusive regimes.




Suppose that the concentration of the product $P_1$ is high. The synthesis of the enzyme $E_2$ is then inhibited, and the product $P_2$ is no longer created. The system may work arbitrarily long in this state.

If for some reason (e.g. a transient fluctuation) the concentration of the product $P_2$ increases, the synthesis of the enzyme $E_1$ and its product $P_1$ will be repressed. Hence, the system will switch to another regime of work, in which the syntheses of $E_2$ and $P_2$ are prevalent.

In order to construct a mathematical model of a trigger, we will follow the changes in concentrations of the products of enzymatic reactions. Their decrease is due to the utilization of the products by the cell, while the increase of their concentrations comes from the synthesis, described by a constant rate of the enzymatic activity:

$$\frac{dP_1}{dt} = C_1 - \kappa_1 P_1$$

$$\frac{dP_2}{dt} = C_2 - \kappa_2 P_2$$

The enzymatic activity may be inhibited by the presence of molecules of the 'other' type of co-repressor. Therefore we will use the following formula for the growth rate of the products:

$$C_1 = \frac{A_1}{B_1 + P_2^m}$$

$$C_2 = \frac{A_2}{B_2 + P_1^n}$$

These relations reflect the ***cross co-repression***, i.e. the inhibiting effect of one of the products on the growth rate of the other one. In this case, parameters ***B*** correspond to the inhibition constants. Coefficients ***A*** are complex functions of many kinetic processes; they depend on the activity of the RNA polymerase, on the rate of ribosomal processes and on concentrations of substrates ***S***. The substrates reflect the level of nutrients in a cell and constitute the only way of exerting an external influence on a cell. Hence, parameters ***A*** reflect generally the rate of cellular metabolism.

The exponents ***m*** and ***n*** define the order of repression, i.e. the number of molecules of each product ***P*** necessary for the efficient repression to occur. If one molecule is sufficient, ***m=n=1***, etc.

First, let's consider a symmetric system, in which $A_1 = A_2 = A$, $B_1 = B_2 = B$, $\kappa_1 = \kappa_2 = \kappa$ and $m = n$:

$$\frac{dP_1}{dt} = \frac{A}{B + P_2^n} - \kappa P_1$$
$$\frac{dP_2}{dt} = \frac{A}{B + P_1^n} - \kappa P_2$$

Let's introduce dimensionless variables:

$$x = \frac{P_1}{B^{\frac{1}{n}}}, \qquad y = \frac{P_2}{B^{\frac{1}{n}}}, \qquad \tau = \kappa t, \quad \text{and} \quad \overline{\mathrm{A}} = \frac{\mathrm{A}}{\kappa \mathrm{B}^{1+\frac{1}{n}}}$$

The original system of equations is transformed to the following form:

$$\frac{dx}{d\tau} = \frac{\overline{A}}{1 + y^n} - x$$
$$\frac{dy}{d\tau} = \frac{\overline{A}}{1 + x^n} - y$$

The stationary states can be found in the usual way:

$$\left.\begin{aligned} 0 &= \frac{\overline{A}}{1+\overline{y}^n} - \overline{x} \\ 0 &= \frac{\overline{A}}{1+\overline{x}^n} - \overline{y} \end{aligned}\right\} \quad \Rightarrow \quad \frac{\overline{A}\left(1+\overline{x}^n\right)^n}{\overline{A}^n + \left(1+\overline{x}^n\right)^n} - \overline{x} = 0$$

For $n = 1$:

$$\frac{\overline{A}\left(1+\overline{x}\right)}{\overline{A} + \left(1+\overline{x}\right)} - \overline{x} = 0 \quad \Rightarrow$$

$$x^2 + \overline{x} - \overline{A} = 0 \quad \Rightarrow$$

$$\overline{x}_1 = \frac{1}{2}\left(-1 - \sqrt{4\overline{A}+1}\right), \quad \overline{x}_2 = \frac{1}{2}\left(-1 + \sqrt{4\overline{A}+1}\right)$$

Note the following:

$$\overline{A} > 0, \qquad \sqrt{4\overline{A}+1} > 1$$
$$\Downarrow$$
$$\overline{x}_1 < 0, \qquad \overline{x}_2 > 0$$

The root corresponding to the negative value of the product's concentration has no physical sense. Consequently, there is only one stationary state in this system. Clearly, such a system cannot be a trigger, because switching requires at least two stationary states.

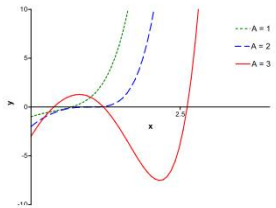
***It follows that the Jacob-Monod schema can become a trigger only when at least two molecules participate in the process of repression (or co-repression)***.

For $n = 2$:

$$\frac{\overline{A}\left(1+\overline{x}^2\right)^2}{\overline{A}^2 + \left(1+\overline{x}^2\right)^2} - \overline{x} = 0 \quad \Rightarrow$$

$$\overline{x}^5 - \overline{A}\overline{x}^4 + 2\overline{x}^3 - 2\overline{A}\overline{x}^2 + \left(\overline{A}^2+1\right)\overline{x} - \overline{A} = 0$$

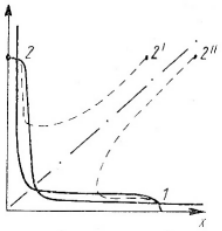
The number of stationary states depends on the parameter $A$:




If $A < A_{cr} = 2$, there is only one stationary state, hence no triggering is possible.

The value of $A_{cr} = 2$ corresponds to a ***bifurcation***. There appear no asymptotic cycles, only multiple unstable states.

For $A > A_{cr}$, there appear three stationary states. Two of them are stable nodes and the third one is a saddle. The phase portrait is presented in the figure below. ***This system may work as a trigger***. The separator passing through the saddle shows the border between the two regions corresponding to the two regimes of the trigger's work.

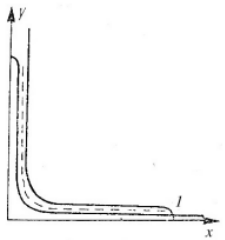



***How to force a transition from one regime to the other?***

Basically, in two ways. First, we may increase the concentration of one of the products directly inside the cell. It corresponds to moving the system to another state, e.g. from ***2*** to ***2'***. If the dose is insufficient, the system will revert to its original state. If the separator will be crossed, as in ***2''***, it will follow a trajectory leading to the new state. From the point of view of biology, we may call this method ***specific***, because it requires adding specific enzymes to the cells.

Another way of affecting the system is to change the system's parameters, e.g. by reducing the supply of one of the substrates. This method is called ***non-specific***; it is easier to control the concentration of substrates than the content of specific substances within a cell. The system's phase portrait is thus deformed. Let's look at the details.

Affecting a substrate's concentration is equivalent to introducing asymmetry in the system. So far, we have dealt with symmetric systems ($A_1 = A_2 = A$, etc). But the general case is more interesting. An example of an asymmetric phase portrait is shown below.




The saddle point merges with one of the stable nodes and they both disappear (e.g. when $A_1$ remains constant while $A_2$ decreases).

When the concentration of one of the substrates is modified, the phase portrait may look as in the figure above and the system will tend to the only stationary point left. When the system's state is close to the point ***1***, the substrate's original concentration (and thus $A_2$) can be restored. The phase portrait will change back to normal, but the system will remain in the new state. Hence, a switching has occurred.

A trajectory corresponding to such a case is shown in the figure (dashed line). As can be seen, it passes close to the isoclines. It means that initially the concentrations of the metabolites corresponding to the old state decrease, and only then those of the new state increase. Between the two regimes there is a ***lag period***, during which a cell is in neither of the regimes.

To go back to the original problem – ***the differentiation of cells*** – we may make analogies between evolution of a fertilized egg and the parameters of a trigger system.

The initial, rapid division of the original cell corresponds to the state, described by a single stationary point. It is equivalent to the phase portrait obtained for $A < A_{cr}$. As the metabolism of the ensemble of cells accelerates, the value of $A$ gradually increases. At some point it will reach the critical value $A_{cr}$ and thus it will find itself in the unstable state of bifurcation, in which any factor may induce the actual differentiation.

The process presented above may be perceived as an ***elementary act of differentiation***. A whole organism would require many of these processes. In principle, we may assume that an ensemble of cells which has successfully completed an act of differentiation, becomes a starting point for another process of this kind on a higher level.

The analysis of the trigger model indicates a possibility of ***self-organization of cellular development***. The basic mechanism of self-organization consists in a marked increase of existing stationary states. The final result of self-organization depends mainly on the parameters of the model, i.e. on the factors, determined by properties of enzymes, coded by DNA. Moreover, each of these parameters is a function of properties of various enzymes. Therefore, ***information about each act of differentiation is coded in many different structural genes***.

This kind of information, dispersed among many different structural elements, may be regarded as ***distributed coding***.

## 9.10 A Mathematical Model for Lead in Mammals

*While lead interacts differently with the various tissues of the body, as a first approximation we need only distinguish three tissue types: bone, blood, and the other soft tissue of the body. Bone tends to take up lead slowly but retain it for very long periods of time, in contrast to soft tissue, other than blood, in which the turnover of lead is much quicker. Blood is the transport agent of the metal. The disposition of lead in the body can be followed as a three-compartment system by tracking its movement into and out of these three tissue types. In this section we analyze such a model proposed by Rabinowitz, Wetherill, and Kopple.*

*The uptake and movement of lead can be modeled by the use of compartments.*

The activity of lead in the body depends on the tissue in which it is located (recall the end of Section 9.1). To construct a mathematical model for the movement of lead, at least three distinct tissue types must be recognized: bone, blood, and soft tissue (other than blood).[15] These will be our mathematical compartments. Lead enters the system by ingestion and through the skin and lungs. These intake paths usher the substance to the blood. From the blood, lead is taken up by bone and by soft tissue. This uptake is reversible: Lead is also released by these organic reservoirs back to the blood. However, especially for bone, lead's half-life in that tissue is very long. Lead can be shed from the body via the kidneys from the blood and to a lesser extent, through hair. Thus blood is the main conduit through which lead moves among our compartments.

To begin the model, let compartment 1 be the totality of the victim's blood, compartment 2 the soft tissue, and compartment 3 the skeletal system. We must also treat the environment as another compartment to account for lead intake and elimination; we designate it as compartment 0. Let $x_i$, $i = 1, \ldots, 3$, denote the amount of lead in compartment $i$ and let $a_{ij}$, $i = 0, \ldots, 3$, $j = 1, \ldots, 3$, denote the rate of movement of lead *to* compartment $i$ *from* compartment $j$. The product $a_{ij}x_j$ is the rate at which the amount of lead increases in compartment $i$ due to lead in compartment $j$. There is no reason that $a_{ij}$ should equal $a_{ji}$, and as noted above, the rate of movement from blood to bone is very different from the reverse rate. The units of the $a_{ij}$s are per day.

$$\begin{aligned}
\frac{dx_1}{dt} &= -(a_{01} + a_{21} + a_{31})x_1 + a_{12}x_2 + a_{13}x_3 + I_L(t), \\
\frac{dx_2}{dt} &= a_{21}x_1 - (a_{02} + a_{12})x_2, \\
\frac{dx_3}{dt} &= a_{31}x_1 - a_{13}x_3.
\end{aligned} \tag{9.10.1}$$

In words, the first equation, for example, says that lead leaves the blood for the environment, soft tissue, and bone at a rate in proportion to the amount in the blood; lead enters the blood from the soft tissue and bone in proportion to their respective amounts; and lead enters the blood from the environment according to $I_L(t)$. The algebraic sum of these effects is the rate of change of lead in the blood.

$$\mathbf{X}' = A\mathbf{X} + \mathbf{f}. \tag{9.10.2}$$

Here $\mathbf{X}$ is the vector of $x$s, $\mathbf{f}$ is the vector

$$\mathbf{f} = \begin{bmatrix} I_L(t) \\ 0 \\ 0 \end{bmatrix},$$

and $A$ is the $3 \times 3$ matrix

$$A = \begin{bmatrix} -(a_{01} + a_{21} + a_{31}) & a_{12} & a_{13} \\ a_{21} & -(a_{02} + a_{12}) & 0 \\ a_{31} & 0 & -a_{13} \end{bmatrix}.$$

From (2.4.12), the solution is

$$\mathbf{X} = e^{At}\mathbf{X}_0 - [I - e^{At}]A^{-1}\mathbf{f} = e^{At}[\mathbf{X}_0 + A^{-1}\mathbf{f}] - A^{-1}\mathbf{f}.$$

*A study on human subjects.*

Rabinowitz, Wetherill, and Kopple studied the lead intake and excretion of a healthy volunteer living in an area of heavy smog. Their work is reported in [3] and extended by Batschelet, Brand, and Steiner in [11]. (See also [12].) The data from this study were used to estimate the rate constants for the compartment model (9.10.1). Lead is measured in micrograms and time in days. For example, the rate 49.3 is given below as the ingestion rate of lead in micrograms per day, and the other coefficients are as given in Table 9.10.1.

**Table 9.10.1.** Lead exchange rates.

| **coefficients** | $a_{01}$ | $a_{12}$ | $a_{13}$ | $a_{21}$ | $a_{02}$ | $a_{31}$ | $I_L$ |
|---|---|---|---|---|---|---|---|
| **value** | 0.0211 | 0.0124 | 0.000035 | 0.0111 | 0.0162 | 0.0039 | 49.3 |

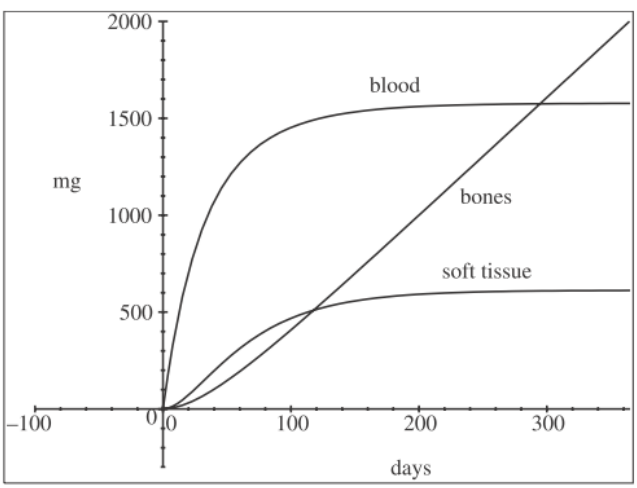



**Fig. 9.10.1.** Solutions for (9.10.1).

$$X_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

$$\mathbf{X} = e^{At}\mathbf{X}_0 - [I - e^{At}]A^{-1}\mathbf{f} = e^{At}[\mathbf{X}_0 + A^{-1}\mathbf{f}] - A^{-1}\mathbf{f}$$

MATLAB
```
> [evect,eval]=eig(A)
```

This computation yields

$$-0.0447, \qquad -0.02, \quad \text{and} \quad -0.00003.$$

MATLAB
```
> leadlim=-AinvF
```

yields

$$-A^{-1}\mathbf{f} = (1800, 698, 200582), \quad \text{where } \mathbf{f} = \left(\frac{439}{10}, 0, 0\right).$$

## 5. Direction fields.

To sketch a direction field for a first order differential equation with MATLAB, the equation must be in **normal form.** A first order differential equation is in normal form if it looks like this

$$y' = f(t, y).$$

To sketch a direction field, we use the MATLAB functions **meshgrid** and **quiver.** Briefly, **meshgrid** creates a grid of points in the $(t, y)$-plane, and **quiver** plots little vectors at each point, whose directions are determined by the righthand side of the differential equation.

The **basic** syntax is something like this:

```
>> [T Y] = meshgrid(minT:step:maxT, minY:step:maxY);
>> dY = f(T,Y);
>> dT = ones(size(dY));
>> quiver(T,Y, dT, dY);
```


```
>> [T Y] = meshgrid(minT:step:maxT, minY:step:maxY);
>> dY = f(T,Y);
>> dT = ones(size(dY));
>> quiver(T,Y, dT, dY);
```


**Explanation:**

a. **meshgrid** creates a uniformly spaced grid of points in the rectangle

$$\{(T, Y) : minT \leq T \leq maxT, \quad minY \leq Y \leq maxY\}$$

and assigns the horizontal coordinates of the points to $T$ and the vertical coordinates to $Y$. The spacing is determined by the parameter 'step'.

b. The command 'dY = f(T,Y)' computes the matrix of slopes of the vectors attached to each point in the grid. Note that you need to type in an actual function of $T$ and $Y$ here (not just write 'f(T,Y)').

c. This creates a matrix of 1s of the same dimension as dY.

d. The vector that the 'quiver' command plots at the point (T,Y) in the grid will be parallel to the vector (dT,dY) = (1,dY), giving it the correct slope. Quiver automatically scales the vectors so that they do not overlap.

I'll use the syntax above to sketch a direction field for the differential equation $y' = \cos 2t - y/t$ in the rectangle $\{(t, y) : 0 \le t \le 8,\ -4 \le y <\le 4\}$:

```
>> [T Y]=meshgrid(0:0.2:4,-4:0.2:4);
>> dY=cos(2*T)-Y./T;
>> dT=ones(size(dY));
>> quiver(T, Y, dT, dY);
```

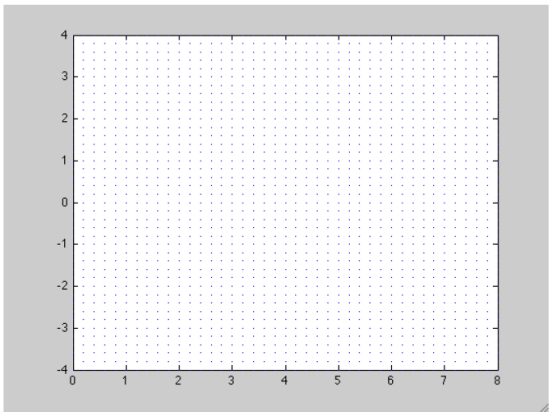



Figure 8: First try at direction field for $y' = \cos 2t - t/y$.

The (unsatisfying) result is in Figure 8, above. The problem lies with the automatic scaling feature of quiver, and the fact that the vectors in the direction field above have vastly different lengths. To fix this problem, we can scale all the vectors to have unit length, by dividing each one by its length. Inserting the command

```
>> L=sqrt(1+dY.^2);
```

before the quiver command in the sequence above, and then changing the quiver command to

```
>> quiver(T, Y, dT./L, dY./L)
```

produces the output in Figure 9.

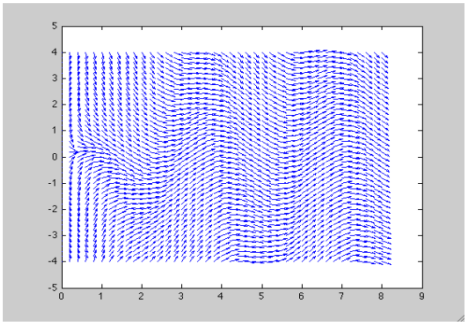



Figure 9: Second try at direction field for $y' = \cos 2t - t/y$.

This one looks like a direction field, but there are still things to fix, e.g. the white space around the direction field and the fact that the vectors are now overlapping in places. The pair of commands

```
>> quiver(T, Y, dT./L, dY./L, 0.5)
>> axis tight
```

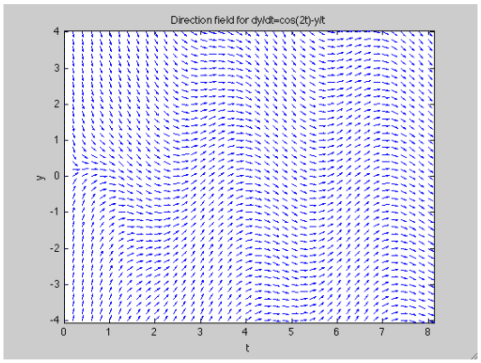



Figure 10: Rescaled direction field without annoying white space

produces the more pleasing direction field in Figure 10, where the vectors have been scaled to half their former length with the optional argument '0.5' in the quiver command and the white space is removed with the 'axis tight' command.

Since quiver is a plotting command many (if not all) of the optional plotting arguments and commands can be used in conjunction with quiver. You can add labels to the axes, titles to the figures, change colors, etc. For example, the commands

```
>> xlabel 't', ylabel 'y';
>> title 'Direction field for dy/dt=cos(2t)-y/t';
>> quiver(T, Y, dT./L, dY./L, 0.5,'r'), axis tight
```

produce the red direction field, complete with axis labels and title in Figure 11.

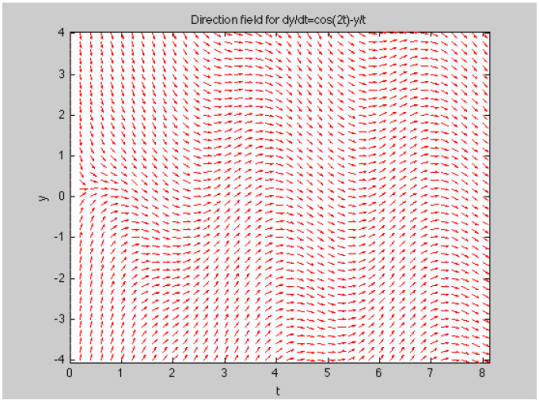



Figure 11: Red direction field for $y' = \cos 2t - t/y$.

**Exemple d'un programme qui affiche le champ vectoriel pour la fonction 'logistic' et puis y ajoute trois trajectoires différentes, pour les points initiaux différents.**

```
t=0:0.5:8;
x=0:0.2:2;

function dx=logistic(x,t)
  dx=x-x^2;
endfunction

nt=length(t);
nx=length(x);
V1=zeros(nt,nx);
V2=zeros(nt,nx);

for i=1:nt
  for j=1:nx
    V1(i,j) = 1;
    V2(i,j) = logistic(x(j),t(i));
  endfor
endfor
```

```
Vnorm=sqrt(V1.^2+V2.^2);
V1=V1./Vnorm;
V2=V2./Vnorm;
scalefactor = 0.3;
figure(1) ;
quiver(t,x,V1',V2',scalefactor)
axis([0,8,0,2]);
xlabel('time t'); ylabel('population p');
grid on

t=0:0.1:8;
x1=lsode('logistic',0.2,t);
x2=lsode('logistic',0.4,t);
x3=lsode('logistic',2.0,t);
hold on
plot(t,x1,t,x2,t,x3)
axis([0.8,0.2]);
hold off
```

Journal of Computer Science 1 (1): 31-34, 2005
ISSN 1549-3636


# Epidemiological Models Applied to Viruses in Computer Networks

[1]José Roberto Castilho Piqueira, [1]Betyna Fernández Navarro and
[1,2]Luiz Henrique Alves Monteiro
[1]Telecommunications and Control Engineering Department, Politechnic School,
Sao Paulo University,Avenida Prof. Luciano Gualberto, travessa 3 - 158, 05508-900 São Paulo, Brazil
[2]Electrical Engineering Graduate Department, Mackenzie Presbiterian University

**Abstract:** To investigate the use of classical epidemiological models for studying computer virus propagation we described analogies between computer and population disease propagation using SIR (Susceptible-Infected-Removed) epidemiological models. By modifying these models with the introduction of anti-viral individuals we analyzed the stability of the disease free equilibrium points. Consequently, the basal virus reproduction rate gives some theoretical hints about how to avoid infections in a computer network. Numerical simulations show the dynamics of the process for several parameter values giving the number of infected machines as a function of time.

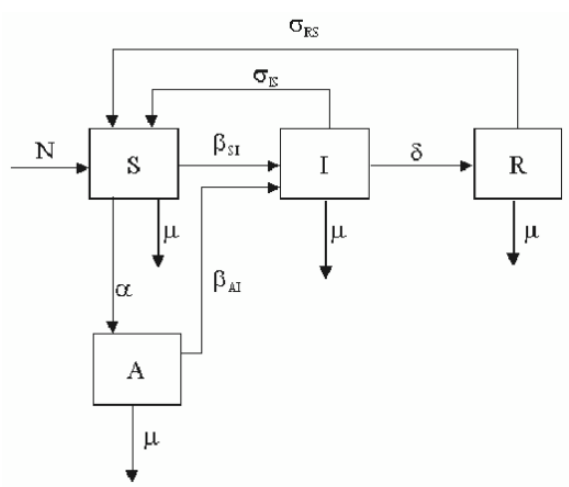



Fig. 1: Susceptible-Antidotal-Infected-Removed Model

$$
\begin{aligned}
\frac{dS}{dt} &= -\alpha SA - \beta_{SI} SI + \sigma_{IS} I + \sigma_{RS} R \\
\frac{dA}{dt} &= \alpha SA - \beta_{AI} A\, I \\
\frac{dI}{dt} &= \beta_{SI} SI + \beta_{AI} AI - \sigma_{IS} I - \delta I \\
\frac{dR}{dt} &= \delta I - \sigma_{RS} R
\end{aligned}
$$

| Parameter | Description |
|---|---|
| $\alpha$ | conversion rate of susceptible computers into antidotal ones |
| $\beta_{SI}$ | Infection rate of new computers |
| $\sigma_{IS}$ | Recovery rate of infected computers |
| $\sigma_{RS}$ | Recovery rate of removed computers with system administrator intervention |
| $\beta_{AI}$ | Infection rate of antidotal computers due to the onset of new virus |
| $\delta$ | Removal rate of infected computers |

Table 4.3: Model parameters

$$
\begin{aligned}
\frac{dS}{dt} &= -\alpha SA - \beta_{SI} SI + \sigma_{IS} I \\
\frac{dA}{dt} &= \alpha SA - \beta_{AI} A I \\
\frac{dI}{dt} &= \beta_{SI} SI + \beta_{AI} AI - \sigma_{IS} I
\end{aligned}
$$

$$
J = \begin{bmatrix} -A\alpha - I\beta_{SI} & -\beta_{SI} S + \sigma_{IS} & -\alpha S \\ I\beta_{SI} & \beta_{SI} S + \beta_{AI} A - \sigma_{IS} & \beta_{AI} I \\ A\alpha & -\beta_{AI} A & \alpha S - \beta_{AI} I \end{bmatrix}
$$

**Problem 63.** *1. Show that $E_1 = (0, N, 0, 0)$ and $E_2 = (N, 0, 0, 0)$ are the two disease-free equilibrium points, where $N = S(0) + A(0) + I(0) + R(0)$.*

*2. Compute the associated Jacobian matrices and compute the eigenvalues. You will notice that both contain a zero eigenvalue. Since $E_2$ has both positive and negative eigenvalues, it is a saddle, and thus unstable.*

*3. The zero eigenvalue for $E_1$ is innocuous because the A-axis is the center manifold. Show that $E_1$ is locally attracting (forgetting about the zero eigenvalue) provided that*

$$R_0 = \frac{\beta_{AI} N}{\sigma_{IS} + \delta} < 1. \tag{4.89}$$

*4. Can you interpret this condition in terms of computer networks? What can a systems administrator do to prevent a virus epidemic on her system?*

M2I BBS

# Modélisation et Simulation de Systèmes Biologiques

## (#10 - Projets)

**Georges Czaplicki, UPS / IPBS-CNRS**

**Tél. : 05.61.17.54.04, email : cgeorge@ipbs.fr**

# Projets

**Project #2**

Suppose that the spruce budworm, in the absence of predation by birds, will grow according to a simple logistic equation of the form

$$\frac{dB}{dt} = rB\left(1 - \frac{B}{K}\right)$$

Budworms feed on the foliage of trees. The size of the carrying capacity, ***K***, would depend on the amount of foliage on the trees. We take it to be constant for this model.

a) Draw graphs for how the population might grow if ***r*** were 0.48 and ***K*** were 15. Use several initial values.

b) Introduce predation by birds into this model in the following manner: suppose that for small levels of worm population there is almost no predation, but for larger levels birds are attracted to this food source. Allow for a limit to the number of worms that each bird can eat. A model for predation by birds might have the form

$$P(B) = a\frac{B^2}{b^2 + B^2}$$

where ***a*** and ***b*** are positive. Sketch the graph for the level of predation of the budworms as a function of the size of the population. Take ***a*** and ***b*** to be 2.

c) A model for the budworm population size in the presence of predation could be modeled

$$\frac{dB}{dt} = rB\left(1 - \frac{B}{K}\right) - a\frac{B^2}{b^2 + B^2}$$

as

To understand the delicacy of this model and the implications for the care that needs to be taken in modeling, investigate graphs of solutions for this model with parameters ***r*** = 0.48, ***a*** = ***b*** = 2, and ***K*** = 15 or ***K*** = 17.

d) Determine the number and type of solutions for each case. Comment on the significance of the results.

P6. Transport de protéines du réticulum endoplasmique à la bordure de la cellule en passant à travers le Golgi

1. Vous représenterez les données expérimentales suivantes où N représente le nombre total de la protéine considérée dans chacun des compartiments cellulaires:

| T (min) | 0 | 10 | 30 | 45 | 60 | 85 |
|---|---|---|---|---|---|---|
| $N_{RE}$ | 96 | 83 | 60 | 33 | 23 | 16 |
| $N_{Golgi}$ | 4 | 10 | 25 | 27 | 1 | 8 |
| $N_{bordure}$ | 0 | 7 | 15 | 40 | 65 | 76 |

2. Vous en déduirez qualitativement des caractéristiques de ce transport.

Il a en fait été proposé que ce transport pouvait être décrit par le système d'équations différentielles suivant :

$$\frac{dN_{RE}}{dt} = -k_{RE}N_{RE} \quad \text{et} \quad \frac{dN_{Golgi}}{dt} = k_{RE}N_{RE} - k_{Golgi}N_{Golgi} \quad \text{et} \quad \frac{dN_{bord}}{dt} = k_{Golgi}N_{Golgi}$$

3. Vous justifierez un tel choix d'équation pour décrire le phénomène de transport (schéma).
4. A partir de la solution analytique trouvée, vous conclurez sur l'adéquation de la description mathématique et les résultats expérimentaux.
5. Vous étudierez ensuite le système suivant d'équations différentielles :

$$\frac{dN_{REint}}{dt} = -k_{REint}N_{REint} \quad \text{et} \quad \frac{dN_{RE}}{dt} = k_{REint}N_{REint} - k_{RE}N_{RE}$$

$$\text{et} \quad \frac{dN_{Golgi}}{dt} = k_{RE}N_{RE} - k_{Golgi}N_{Golgi} \quad \text{et} \quad \frac{dN_{bord}}{dt} = k_{Golgi}N_{Golgi}$$

et déduirez les constante ($k_{REint}$, $k_{RE}$ et $k_{Golgi}$) par ajustement non linéaire.

6. Conclusion et lien avec données expérimentalement accessibles.

1.4. Harvesting problem.

Let us consider a population growing according to a logistic dynamics. Let assume that a constant effort $E$ of fishing[1] so that the yield per unit time is $qEx$, where $x$ is population size $q$ is a coefficient denoting the return to effort.

(a) Write down the differential equation for $x(t)$ which translate these assumptions; find its equilibria and the asymptotic behaviour of solutions, according to parameter values.

(b) Let assume that the unit price at which the fish is sold is $p$, and that the cost of fishing is proportional (through a coefficient $c$) to the effort $E$. Let assume that an enlightened dictator wants to set $E$ at the value that maximizes the gain (= revenue − cost) when the population is at its asymptotically stable equilibrium. Find the value of $E$ and the corresponding equilibrium value for $x$.

(c) Economic theory predicts that, for an open access fishery, the effort $E$ will in the long run reach the value at which the gain is equal to 0. Find the value of $E$ and the corresponding equilibrium value of $x$; compare them (i.e, find, if they are greater or smaller) than the previous case.

(d) Let assume that the government taxes at a percentage $\rho$ the gains obtained by fisheries. How does this affect the results obtained with open-access fishery?

[1] or hunting, or harvesting

(e) Let assume that the government taxes according to how much has been fished $Y$. Let us consisre two separate cases: a constant fraction $\rho Y$, or a progressive tax $\tau(Y)$ given by the formula

$$\tau(Y) = \begin{cases} 0 & \text{se } Y \leq Y_0 \\ \rho(Y - Y_0) & \text{se } Y > Y_0 \end{cases}$$

Which are the results of these regulations?

(f) Let us assume that the dynamics of $x$ be described, in absence of fishing, by the generalized logistic equation

$$x'(t) = rx(t)\left(1 - \left(\frac{x(t)}{K}\right)^{\alpha}\right). \qquad \alpha > 0$$

How do previous results change?

OPEN ACCESS Freely available online PLoS COMPUTATIONAL BIOLOGY

# Living on Three Time Scales: The Dynamics of Plasma Cell and Antibody Populations Illustrated for Hepatitis A Virus

**Mathieu Andraud[1]*, Olivier Lejeune[1,2], Jammbe Z. Musoro[3,4], Benson Ogunjimi[1], Philippe Beutels[1], Niel Hens[1,3]**

**1** Centre for Health Economics Research and Modelling of Infectious Diseases (CHERMID), Vaccine & Infectious Disease Institute (VAXINFECTIO), University of Antwerp, Antwerp, Belgium, **2** The SYMBIOS Center, Division of Mathematics, University of Dundee, Dundee, United Kingdom, **3** Interuniversity Institute of Biostatistics and Statistical Bioinformatics, Hasselt University, Diepenbeek, Belgium, **4** Academic Medical Center, University of Amsterdam, Amsterdam, The Netherlands

## Abstract

Understanding the mechanisms involved in long-term persistence of humoral immunity after natural infection or vaccination is challenging and crucial for further research in immunology, vaccine development as well as health policy. Long-lived plasma cells, which have recently been shown to reside in survival niches in the bone marrow, are instrumental in the process of immunity induction and persistence. We developed a mathematical model, assuming two antibody-secreting cell subpopulations (short- and long-lived plasma cells), to analyze the antibody kinetics after HAV-vaccination using data from two long-term follow-up studies. Model parameters were estimated through a hierarchical nonlinear mixed-effects model analysis. Long-term individual predictions were derived from the individual empirical parameters and were used to estimate the mean time to immunity waning. We show that three life spans are essential to explain the observed antibody kinetics: that of the antibodies (around one month), the short-lived plasma cells (several months) and the long-lived plasma cells (decades). Although our model is a simplified representation of the actual mechanisms that govern individual immune responses, the level of agreement between long-term individual predictions and observed kinetics is reassuringly close. The quantitative assessment of the time scales over which plasma cells and antibodies live and interact provides a basis for further quantitative research on immunology, with direct consequences for understanding the epidemiology of infectious diseases, and for timing serum sampling in clinical trials of vaccines.

OPEN ACCESS Freely available online PLoS COMPUTATIONAL BIOLOGY

# Multi-Scale Modeling of HIV Infection *in vitro* and APOBEC3G-Based Anti-Retroviral Therapy

**Iraj Hosseini, Feilim Mac Gabhann***

Institute for Computational Medicine and Department of Biomedical Engineering, Johns Hopkins University, Baltimore, Maryland, United States of America

## Abstract

The human APOBEC3G is an innate restriction factor that, in the absence of Vif, restricts HIV-1 replication by inducing excessive deamination of cytidine residues in nascent reverse transcripts and inhibiting reverse transcription and integration. To shed light on impact of A3G-Vif interactions on HIV replication, we developed a multi-scale computational system consisting of intracellular (single-cell), cellular and extracellular (multicellular) events by using ordinary differential equations. The single-cell model describes molecular-level events within individual cells (such as production and degradation of host and viral proteins, and assembly and release of new virions), whereas the multicellular model describes the viral dynamics and multiple cycles of infection within a population of cells. We estimated the model parameters either directly from previously published experimental data or by running simulations to find the optimum values. We validated our integrated model by reproducing the results of *in vitro* T cell culture experiments. Crucially, *both* downstream effects of A3G (hypermutation and reduction of viral burst size) were necessary to replicate the experimental results *in silico*. We also used the model to study anti-HIV capability of several possible therapeutic strategies including: an antibody to Vif; upregulation of A3G; and mutated forms of A3G. According to our simulations, A3G with a mutated Vif binding site is predicted to be significantly more effective than other molecules at the same dose. Ultimately, we performed sensitivity analysis to identify important model parameters. The results showed that the timing of particle formation and virus release had the highest impacts on HIV replication. The model also predicted that the degradation of A3G by Vif is not a crucial step in HIV pathogenesis.

**Exercise 6.2:**
Change the Lotka-Volterra-Model slightly, in assuming logistic population dynamics for the prey alone,

$$\begin{aligned} \frac{d}{dt}N &= aN(1-N) - bNP \\ \frac{d}{dt}P &= cNP - dP \end{aligned}$$

**(a)** Compute the stationary points.
**(b)** Consider parameters, s.t. a stationary solution exists with predator and prey present. Show that this stationary point is locally stable, i.e. the eigenvalues of the linearization have negative real part.

P6. Compétition entre lynxs et lièvres

Les populations de lynx (**y**) et de lièvres (**x**) évoluent selon le modèle suivant :

$$\frac{dx}{dt} = rx\left[1 - \frac{x}{K}\right] - b\frac{x}{\alpha + x}y \quad \text{et} \quad \frac{dy}{dt} = -cy + D\frac{x}{\alpha + x}y$$

Où r, K, b, α, c, D sont des constantes positives

1. Vous interpréterez biologiquement les termes apparaissant dans ces équations en donnant les dimensions des paramètres.
2. Par une analyse graphique, vous déterminerez les équilibres rn précisant leur stabilité ainsi que la dynamique de ces deux populations. Vous pourrez choisir comme paramètres :

| r | K | b | c | D | α |
|---|---|---|---|---|---|
| 1,2 | 10 | 1 | 0,1 | 0,2 | 0,5 ou 3 ou 4 ou 7 |

3. Vous en déduirez la dynamique du système lorsque le paramètre a varie au cours du temps selon :
$$\frac{d\alpha}{dt} = -h(\alpha - \alpha_m)$$
4. Conclusion

## Project #3

Imagine a *three-species predator-prey* problem, which we identify with grass, sheep and wolves. The grass grows according to a logistic equation in the absence of sheep. The sheep eat the grass and the wolves eat the sheep. Each on its own, the populations of both sheep and wolves decrease with a constant (negative) *per capita* growth rate.

For practical reasons (graphs) assume the following parameter values:

- growth rates: 2 for grass, -1 for sheep and wolves
- carrying capacity for grass: 2
- encounter benefits: -1 for grass because of sheep, +2 for sheep because of grass, -1 for sheep because of wolves, +1 for wolves because of sheep.

a) Write and analyze the system of equations (stationary points, stability, trajectories…).
b) What would be the steady state of grass, with no sheep or wolves present?
c) What would be the steady state of sheep and grass, with no wolves present?
d) What is the revised steady state with wolves present? Does the introduction of wolves benefit the grass?

## Project #4

(based on project #3)

The following is a *predator-prey model with child care*. Suppose that the prey is divided into two classes: of young and adults. Suppose that the young are protected from predators. Assume the young increase proportional to the number of adults and decrease due to death or to moving into the adult class. The number of adults is increased by the young growing up and decreased by natural death and predation. Finally, the predators die naturally and benefit from encounters with adults.

Write and analyze the equations (stationary points, stability, trajectories…).

**Note:**
For practical purposes (graphs) use the following parameter values:

- for the young: birth rate = 2, mortality rate = ½, growing up = ½
- for the adults: natural death rate = ½, predation mortality = 1
- for the predators: death rate = 1, predation benefits = 1.

**Exercise 2:** (Self intoxicating population) (4)

Some populations (such as algae and bacteria) produce waste products, which in high concentrations are toxic to the population itself. A typical mathematical model for a population $n(t)$ and a toxic waste product $y(t)$ is

$$\begin{aligned} \dot{n} &= (\alpha - \beta - Ky)n \\ \dot{y} &= \gamma n - \delta y \end{aligned}$$

with $\alpha, \beta, \gamma, \delta, K > 0$.

1. Explain each term in the above model and sketch an arrow diagram.
2. Find the nullclines, the steady states and sketch a phase portrait and the vectorfield.
3. Linearize the system at the steady states and determine their stability. Find the regions in parameter space such that the nontrivial (coexistence) equilibrium is a node or a spiral.
4. Sketch some trajectories for the case of $\delta < 4(\alpha - \beta)$ and explain what you see in terms of biology.
5. Consider the case of high dilution $\delta >> 1, \gamma/\delta < \infty$.

*Chapter 4*

# WHEN ZOMBIES ATTACK!: MATHEMATICAL MODELLING OF AN OUTBREAK OF ZOMBIE INFECTION

***Philip Munz*[1]*, *Ioan Hudea*[1]†, *Joe Imad*[2]‡, *Robert J. Smith?*[3]§**
[1]School of Mathematics and Statistics, Carleton University,
1125 Colonel By Drive, Ottawa, ON K1S 5B6, Canada
[2]Department of Mathematics, The University of Ottawa,
585 King Edward Ave, Ottawa ON K1N 6N5, Canada
[2]Department of Mathematics and Faculty of Medicine, The University of Ottawa,
585 King Edward Ave, Ottawa ON K1N 6N5, Canada

**Abstract**

Zombies are a popular figure in pop culture/entertainment and they are usually portrayed as being brought about through an outbreak or epidemic. Consequently, we model a zombie attack, using biological assumptions based on popular zombie movies. We introduce a basic model for zombie infection, determine equilibria and their stability, and illustrate the outcome with numerical solutions. We then refine the model to introduce a latent period of zombification, whereby humans are infected, but not infectious, before becoming undead. We then modify the model to include the effects of possible quarantine or a cure. Finally, we examine the impact of regular, impulsive reductions in the number of zombies and derive conditions under which eradication can occur. We show that only quick, aggressive attacks can stave off the doomsday scenario: the collapse of society as zombies overtake us all.