

---

## M 3 : Applications EDO avec *Mathematica*

---

### Préambule

Ce préambule constitue un petit manuel de référence et d'autoformation pour calculer des EDO avec *Mathematica*. Il aborde les principales difficultés et questions, et doit être lu avant d'aborder la séance de TPTD2.

---

## I. Objectif général et Présentation

L'objectif général de M3 est de traiter efficacement les EDO à l'aide de *Mathematica*.

*Mathematica* est conçu initialement comme un système (informatique) pour faire des Mathématiques. Cette conception a deux conséquences pratiques.

Premièrement, c'est pour cela que le langage Wolfram est très proche du langage mathématique, mais aussi qu'il est plus précis, puisque c'est aussi un système informatique, et donc technique. Cette exigence de précision, ou de rigueur technique est souvent déconcertante pour les débutants de *Mathematica*, avant de réaliser qu'elle répond clairement à la nécessité de disposer de notations non ambiguës. Nous invitons le lecteur à reprendre le chapitre M1 qui résume les principaux éléments généraux de ces notations.

Deuxièmement, c'est pour cela que *Mathematica* cherche non seulement à traiter des problèmes mathématiques "papier crayon", mais aussi plus largement à étendre le domaine des mathématiques aux problèmes intraitables sans aide informatique. Pour autant, pouvoir faire plus ne signifie pas que des traitements "papier-crayon" apparemment simples sont tous abordables directement dans *Mathematica*.

Pour traiter les EDO, nous allons d'abord aborder plusieurs points techniques indispensables dans ce préambule :

1. La manipulation des matrices et le calcul matriciel,
2. Les représentations graphiques,
3. Les champs de vecteurs
4. La gestion des variables d'EDO, et de leurs solutions

Avec le chapitre Exercices qui suit, nous reprendrons ces points et nous aborderons les applications.

---

## II. Calcul matriciel

### ■ A. Définitions des matrices \*

Nous allons aborder les aspects techniques indispensables du calcul matriciel à maîtriser avec *Mathematica*.

#### ■ 1. Définition d'une matrice

**Déf.** Une matrice réelle est un tableau rectangulaire de nombres réels notée entre grandes parenthèses ou crochets :

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \dots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix} \text{ où } a_{i,j} \in \mathbb{R} \text{ pour } 1 \leq i \leq m, 1 \leq j \leq n$$

est une matrice à  $m$  lignes et  $n$  colonnes, plus communément dite de taille  $m \times n$ .

Plus généralement, la matrice peut être réelle ou complexe, si  $a_{i,j} \in \mathbb{R}$  ou  $\mathbb{C}$ .

Attention : on indique toujours en premier le nombre de lignes et en second celui des colonnes. De même le coefficient  $a_{i,j}$  est indexé d'abord par l'indice  $i$  de sa ligne, puis l'indice  $j$  de sa colonne.

Notation : traditionnellement en mathématiques, les matrices sont souvent notées par des majuscules en gras pour les distinguer des autres objets.

Une matrice sert à réécrire très naturellement un système d'équations linéaires de façon très concise. De la même façon, elle permet de réécrire un système d'équations différentielles linéaires.

## ■ 2. Définition Mathematica d'une matrice

**Déf.** Une matrice est une liste de listes qui correspond à un tableau rectangulaire de nombres (réels ou complexes). Les listes sont notées entre accolades.

D'un point de vue strictement technique, c'est un objet pour lequel on a : `MatrixQ[objet]==True`

Par exemple, `{{a,b},{c,d}}` est bien une matrice :

`MatrixQ[{{a,b},{c,d}}]==True`

On peut l'afficher sous forme traditionnelle avec `MatrixForm` :

`(matT = {{a,b},{c,d}}) // MatrixForm`

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

## ■ 3. Opérations matricielles

On peut faire toutes les opérations matricielles avec la forme brute, `{{a,b},{c,d}}` d'une matrice, et celle d'un vecteur  $u = \{ux, uy\}$  :

`matT.{ux,uy}=={{a,b},{c,d}}.{ux,uy}=={a ux+b uy,c ux+d uy}`

Au passage, on observe que le produit scalaire de deux vecteurs, et le produit matriciel (deux matrices ou une matrice et un vecteur) est noté de la même façon dans *Mathematica* :

`{ux,uy}.{vx,vy}==Dot[{ux,uy},{vx,vy}]==ux vx+uy vy`

Mais il n'est pas possible de faire ce calcul avec la mise en forme des vecteurs ou des matrices. C'est habituel dans *Mathematica* car les mises en forme ne sont généralement pas utilisables : un tableau n'est pas sa mise en forme. Ainsi, les expressions donnent bien les belles mises en forme matricielle (comme attendu), mais ne donnent pas de résultat (ici  $ux = 1$  et  $uy = 0$ ) :

`({{a,b},{c,d}} // MatrixForm).({1,0} // MatrixForm)`

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Par contre, il est possible de définir directement des matrices (comme expliqué dans la sous-section d'exercice qui suit), et d'effectuer des calculs matriciels :

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} == \begin{pmatrix} a \\ c \end{pmatrix} == \{a, c\}$$

## ■ → B. Opérations de base sur des matrices \*

Pour saisir une matrice, le plus simple est d'utiliser la suite de menus :

Insert ▶ Create Table/Matrix/ ▶ New

ou encore les raccourcis clavier indiqués dans ce dernier menu.

On peut aussi utiliser la palette :

Palettes ▶ Basic Math Assistant

### ■ → 1) Saisir les matrices suivantes :

$$\mathbf{matM} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \text{ et } \mathbf{matN} = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$$

On obtient :

$$\mathbf{matM} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}; \mathbf{matN} = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix};$$

### ■ → 2) Calculez la somme matricielle : $\mathbf{matM} + 2 \mathbf{matN}$

$\mathbf{matM} + 2 \mathbf{matN}$

$$\{\{a + 2\alpha, b + 2\beta\}, \{c + 2\gamma, d + 2\delta\}\}$$

On observe que : la matrice obtenue est une liste de listes, directement utilisable pour des calculs, mais que le bel affichage matriciel est perdu.

Utilisez **MatrixForm[]** pour retrouver le bel affichage.

**matM + 2 matN // MatrixForm**

$$\begin{pmatrix} a + 2\alpha & b + 2\beta \\ c + 2\gamma & d + 2\delta \end{pmatrix}$$

Notez qu'il n'est pas possible de calculer à partir de ces beaux affichages, sauf si l'on fait un copier-coller. En effet l'opération suivante ne donne pas d'autre résultat que les formes de départ :

**(matM // MatrixForm) + (2 matN // MatrixForm)**

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} + \begin{pmatrix} 2\alpha & 2\beta \\ 2\gamma & 2\delta \end{pmatrix}$$

Par contre, le copier-coller de ces expressions avec **//MatrixForm** donne bien le résultat recherché avec un bel affichage.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} + \begin{pmatrix} 2\alpha & 2\beta \\ 2\gamma & 2\delta \end{pmatrix} // \text{MatrixForm}$$

$$\begin{pmatrix} a + 2\alpha & b + 2\beta \\ c + 2\gamma & d + 2\delta \end{pmatrix}$$

D'une façon générale, il faut toujours dissocier les opérations matricielles regroupées ci-dessous au moyen de parenthèses, et l'affichage qui suit avec **MatrixForm**, en faisant par exemple :

**(matM = {{a, b}, {c, d}}) // MatrixForm**

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

### ■ → 3) Calculer le produit matriciel : **matM.matN**

**Question :** À quoi correspondent les opérations ci-dessous ?

**Dot[matM,matN]==matM.matN**

**Réponse :** Le produit matriciel est défini à l'aide de la fonction :

**Dot[]** en notation fonctionnelle, ou **.** en notation Infix

$$\text{Dot}\left[\begin{pmatrix} a & b \\ c & d \end{pmatrix}, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\right] == \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} == \begin{pmatrix} a\alpha + b\gamma & a\beta + b\delta \\ c\alpha + d\gamma & c\beta + d\delta \end{pmatrix}$$

### ■ → 4) Calculer avec *Mathematica* : **matM<sup>2</sup>** et **matM.matM**

Calculons avec *Mathematica* : **matM<sup>2</sup>**

**matM<sup>2</sup> // MatrixForm**

$$\begin{pmatrix} a^2 & b^2 \\ c^2 & d^2 \end{pmatrix}$$

Attention : c'est le produit naturel de tableaux qui correspond à la matrice des produits terme à terme, et non le carré d'une matrice.

Calculons avec *Mathematica* : **matM.matM**

**matM.matM // MatrixForm**

$$\begin{pmatrix} a^2 + b c & a b + b d \\ a c + c d & b c + d^2 \end{pmatrix}$$

c'est le produit matriciel qui correspond à la composition de 2 applications linéaires.

### ■ → 5) Conclusions

→ Dans *Mathematica*, les matrices sont un cas particulier de listes de listes.

Comme nous l'avons vu, les opérations II.B.1 ou II.B.2 du type, additions d'un nombre à chacune des valeurs d'un vecteur ou d'une matrice, sont possibles dans *Mathematica* comme opérations sur les listes.

→ En mathématique, ces opérations ne sont pas autorisées, ni utilisées (généralement). On ne peut faire des opérations sur les matrices que si les objets ont des dimensions compatibles.

→ Comme nous l'avons vu, les opérations II.A.4 du type, produit terme à terme de deux listes de listes de même longueur est possible dans *Mathematica* comme opérations sur les listes.

→ En mathématique, cette opération n'est pas autorisée, ni utilisée (généralement).

Le produit matriciel est une opération spécifique aux matrices :

Dans le cas de *Mathematica*, cette opération requiert (à juste titre) un symbole nouveau, le "." entre les matrices, car sinon *Mathematica* effectue le produit terme à terme de deux listes de listes de même longueur.

En résumé, dans ce texte, les matrices et le produit matriciel sont respectivement notés par des majuscules en gras, **A**, **B**, ..., et un espace " " pour respecter la tradition mathématique, tandis qu'elles sont notées par matA, matB, ..., et le symbole "." pour effectuer des calculs dans les cellules Input de *Mathematica*.

### III. Programmation graphique

Relire M2\_4M062UtilisationMMaT.txt.nb, section VI.B Programmation graphique, que nous allons mettre en pratique progressivement ci-dessous.

#### ■ → A. Opérations graphiques et rotations ... \*

##### ■ → 1) Tracer le cercle unité (Circle[]) en rouge

Tracer le cercle unité (**Circle[]**), afficher-le en rouge avec des axes, et avec une taille ImageSize→200

Rappel : construction du type :

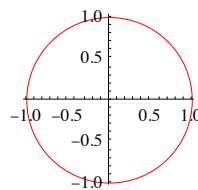
```
Show[
  Graphics[
    { vos directives (couleur) et primitives graphiques Line[], Arrow[], ... }
  ],
  Axes->True, ImageSize->200
]
```

Aller dans l'**aide en ligne** et chercher par exemple "circle". On récupère :

"Circle[{x, y}, r] is a two-dimensional graphics primitive that represents a circle of radius  $r$  centered at the point  $x, y$ ."

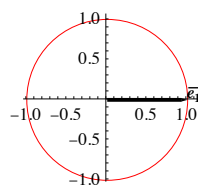
On observe qu'il faut procéder avec un système à plusieurs étapes avec une mise à l'échelle avec **Show[]** :

```
Show[
  Graphics[
    (* Tracé du cercle unité en rouge *) {Red, Circle[{0, 0}, 1]}
  ],
  Axes -> True, ImageSize -> 100
]
```



##### ■ → 2) Tracer le vecteur unité sur l'axe des $x$ (réels), et superposer

```
Show[Graphics[{
  (* Tracé du cercle unité en rouge *) Red, Circle[{0, 0}, 1],
  (* Tracé du vecteur e1 *) Black, Thick, Arrow[{0, 0}, {1, 0}],
  (* Tracé du texte e1 *) Text[Style["e1", Bold], {1 + 0.1, 0.1}]
}], Axes -> True, ImageSize -> 100]
```



(optionnel : rajout de texte  $\vec{e}_1$  à l'extrémité du vecteur, avec la fonction **Text[]**)

- → 3) Afficher le produit  $\begin{pmatrix} \cos[\theta] & -\sin[\theta] \\ \sin[\theta] & \cos[\theta] \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix}$  en bleu (prendre une rotation d'angle  $60^\circ$ )

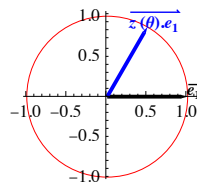
Le calcul donne :

$$\begin{pmatrix} \cos[\theta] & -\sin[\theta] \\ \sin[\theta] & \cos[\theta] \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} == \begin{pmatrix} \cos[\theta] \\ \sin[\theta] \end{pmatrix} == \{\cos[\theta], \sin[\theta]\}$$

Le résultat de ce produit est un vecteur colonne, qui est donc une matrice de deux lignes ne comportant qu'une colonne. Pour être utilisable dans la construction graphique ci-dessous, ce vecteur colonne doit être transformé en un vecteur simple, ou une liste simple au moyen de la fonction **Flatten[]** qui supprime ici un niveau de parenthèses.

$$\text{Flatten}\left[\begin{pmatrix} \cos[\theta] & -\sin[\theta] \\ \sin[\theta] & \cos[\theta] \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right] == \{\cos[\theta], \sin[\theta]\}$$

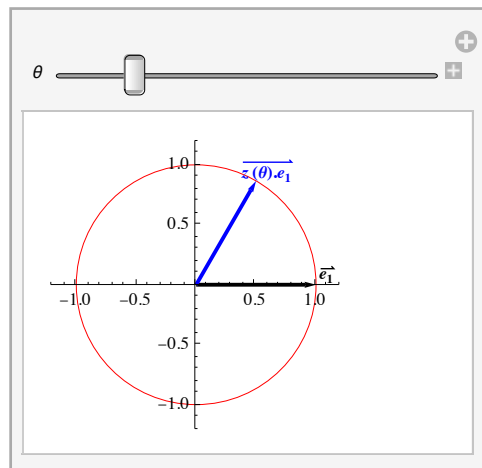
```
Show[Graphics[{
  (* Tracé du cercle unité en rouge *) Red, Circle[{0, 0}, 1],
  (* Tracé du vecteur e1 *) Black, Thick, Arrow[{0, 0}, {1, 0}],
  Text[Style["e1", Bold], {1+0.1, 0.1}],
  (* Tracé du vecteur e1 tourné de pi/3 *)
  Blue, Thick, Arrow[{0, 0}, Flatten[
    {Cos[theta] - Sin[theta], Sin[theta] + Cos[theta]} /. theta -> pi/3]],
  Text[Style["z(theta).e1", Bold], Flatten[
    {Cos[theta] - Sin[theta], Sin[theta] + Cos[theta]} /. theta -> pi/3] + 0.1 /. theta -> pi/3]
} (* Fin de la liste des instructions graphiques *)
], (* Fin de Graphics *)
Axes -> True, ImageSize -> 100] (* Fin de Show *)
```



- → 4) En faire un **Manipulate[]**, cadrer, jouer avec, et conclure.

Un fois le graphique terminé, on peut songer à l'animer interactivement avec **Manipulate[]**.

Il suffit d'encapsuler l'ensemble des instructions avec **Manipulate[]**, et de spécifier  $\theta$ , (en enlevant les anciennes valeurs numériques).



NB : On peut s'amuser à ouvrir la réglette sous le curseur, et à faire tourner automatiquement le vecteur bleu comme une pendule, et aussi jouer avec la stroboscopie...

■ → 5) **Recommandation**

Avant de se lancer dans un **Manipulate[]**, il faut d'abord représenter avec succès un graphique, ou une expression que l'on "Manipulera" en introduisant des variables de contrôle.

## IV. Champ de Vecteurs

Nous allons aborder les aspects pratiques et techniques de champs de vecteurs avec *Mathematica*.

■ **A. Introduction**

■ **1. Champ de vecteurs**

La notion de champ de vecteurs est très pratique.

Par définition, et par construction, le champ de vecteurs d'une application  $f$  est une image de la transformation effectuée pour chaque vecteur  $(x, y)$  du plan : l'image du vecteur  $(x, y)$  par  $f$  est figurée en chaque point  $(x, y)$  du plan.

En pratique, le champ de vecteurs de  $f$  donne en chaque point  $(x, y)$  d'une grille du plan, la valeur de la fonction figurée par un vecteur (avec ou non un facteur d'échelle, calculé pour que les vecteurs sont bien affichés presque partout).

Attention : un vecteur est souvent représenté par une flèche qui a pour origine le point  $O$  de coordonnées  $(0, 0)$  au centre du graphique, tandis qu'ici, chaque vecteur image est placé au point de coordonnées  $(x, y)$  !

■ **2. Éléments de lecture d'un champ de vecteurs**

Les champs de vecteurs d'une application linéaire sont les plus simples possibles. Toutefois leur compréhension nécessite une méthode de lecture. Il faut notamment prendre en compte les notions suivantes :

(i) le vecteur  $\vec{u}$  de coordonnées  $\begin{pmatrix} x \\ y \end{pmatrix}$  est représenté au point de coordonnées  $\begin{pmatrix} x \\ y \end{pmatrix}$  et a une longueur  $\sqrt{x^2 + y^2}$ .

Le long d'une droite vectorielle (passant par l'origine  $O$ )  $\vec{v} = \lambda \vec{u}$  où  $\lambda \in \mathbb{R}$ , les vecteurs images,  $f(\lambda \vec{u}) = \lambda f(\vec{u})$ , sont parallèles, et leur longueur croît linéairement avec la distance entre l'origine et la position. Pour observer l'effet d'une application linéaire à l'aide d'un champ de vecteurs, il suffit donc de regarder les transformations effectuées sur une couronne centrée sur  $O$ .

(ii) Idéalement pour apprécier la transformation effectuée, il faut connaître le vecteur initial et sa transformation. Pour éviter de surcharger les graphes, on ne trace que le vecteur transformé. Il est donc essentiel d'avoir toujours à l'esprit le graphe des vecteurs initiaux, *i.e.* de la transformation identité qui est étudiée en premier.

(iii) Un grand intérêt des champs de vecteurs est de repérer très facilement les droites vectorielles (ou axes) invariants en direction par la transformation (sans même chercher à comprendre ce que fait la transformation !). Il y a souvent un axe invariant en sens alors que pour l'autre le sens est inversé.

(iv) Avec un peu d'habitude, certains graphes (identité, symétrie centrale, rotation, projection, cisaillement) deviennent faciles à comprendre. D'autres, comme les symétries par rapport à un axe, peuvent être plus déconcertants.

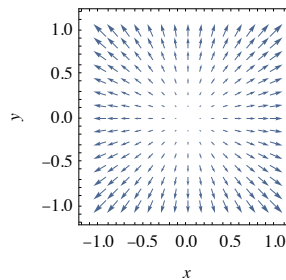
Les champs de vecteurs sont proposés ici comme moyen d'explorer graphiquement les transformations linéaires. Les systèmes d'équations différentielles linéaires, et le champ de vecteurs donne les axes invariants, *i.e.* les cadres de la discussion du système, ainsi qu'une idée de la solution du système, puisqu'il suffit littéralement de suivre les flèches....

■ **B. VectorPlot de l'identité**

■ → **1. Tracé brut du champ de vecteurs de l'identité**

Tracé du champ de vecteurs avec la fonction **VectorPlot[]** de l'identité  $\{x, y\}$  :

```
VectorPlot[
  {x, y}, {x, -1, 1}, {y, -1, 1},
  ImageSize -> {150, 150}, FrameLabel -> {x, y}
]
```



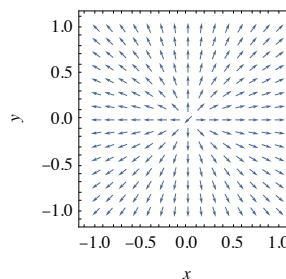
### ■ → 2. Tracé normalisé du champ de vecteurs de l'identité

Même tracé avec l'option **VectorScale** qui contrôle :

la longueur,

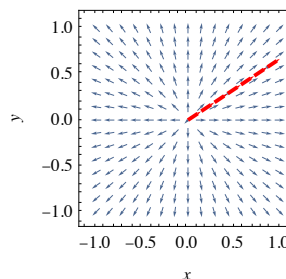
la taille de la pointe des flèches

**None** ici en troisième position force tous les vecteurs à la même taille



### ■ → 3. Idem + 1 ligne de flot

Même tracé avec l'option **StreamPoints** pour suivre le flot (ou les flèches) à partir du point {0.3,0.2}



### ■ → C. VectorPlot de Symétrie Orthogonale[λ, θ] dans un Manipulate

Champ de vecteurs d'une symétrie orthogonale d'angle  $\theta$ , et d'homothétie de rapport  $\lambda$

### ■ → 1. VectorPlot d'une application linéaire simple dans un Manipulate

D'un point de vue code (cf. cellule cachée), on remarque :

l'utilisation très recommandée de la fonction **With[]**,

parce qu'elle est très efficace (deux fois plus rapide que **Module[]**), et

parce qu'elle oblige le programmeur à structurer et à documenter les ordres logiques d'évaluation.

Le code utilise les 2 options de **VectorPlot[]**, **VectorScale** et **StreamPoints** vues précédemment.

Il comporte d'autres éléments de code intéressants :

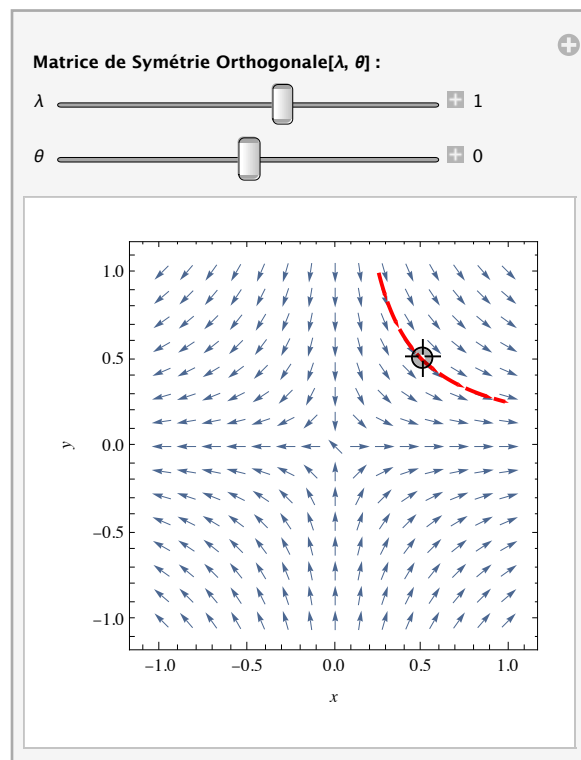
l'objet original **Locator** pour déplacer interactivement un objet dans un graphique,

des aspects d'affichage un peu techniques comme **Control[]**, ou **Style[]**, faciles à comprendre.

```

Manipulate[
  (* I. -----Definitions et calcul des variables ----- *)
  With[{a = λ Cos[θ], b = λ Sin[θ], c = λ Sin[θ], d = -λ Cos[θ]},
    With[{mat =  $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ },
      (* II. -----Affichages ----- *)
      VectorPlot[
        {a * x + b * y, c * x + d * y}, {x, -1, 1}, {y, -1, 1},
        StreamPoints → {{p[[1]], p[[2]]}},
        StreamStyle → {Red, Thick}, VectorScale → {Tiny, Automatic, None},
        ImageSize → {250, 250}, FrameLabel → {x, y}
      ]
    ]],
  (* III. -----Contrôles des variables du Manipulate----- *)
  Style["Matrice de Symétrie Orthogonale[λ, θ] :", Bold],
  Control[{{λ, 1, "λ"}, -5, 5, .01, Appearance → "Labeled"}],
  Control[{{θ, 0, "θ"}, -π, π, .01, Appearance → "Labeled"}],
  {{p, {.5, .5}}, {-1, -1}, {1, 1}, Locator}
]

```



## ■ → 2. Texte et VectorPlot dans deux colonnes adjacentes

Nous reprenons le code précédent en rajoutant au moyen de **Eigensystem[]** le calcul des valeurs propres, et des vecteurs propres de la matrice ainsi que de son déterminant au moyen de **Det[]**.

Les changements de code servent principalement à présenter le texte et le VectorPlot dans deux colonnes au moyen :

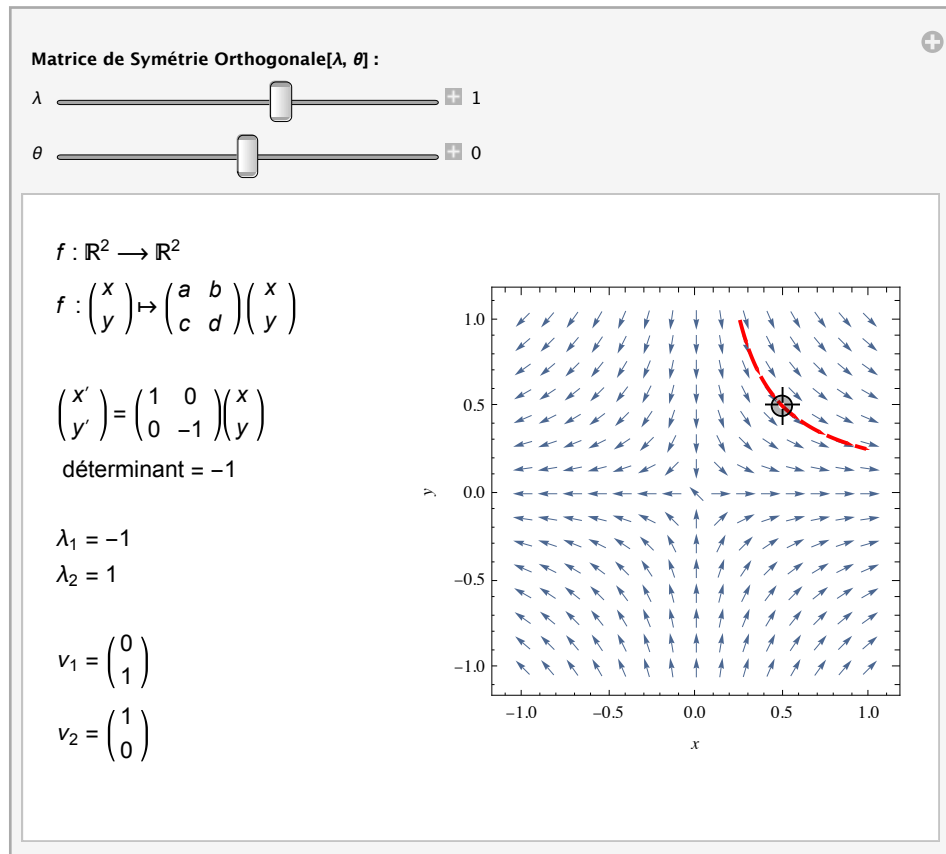
des fonctions **Grid** et **Column** d'organisation graphique en une colonne de texte et le graphique, et de la fonction **TraditionalForm** pour un bel affichage de texte.



```

Manipulate[
  (* I. -----Definitions et calcul des variables ----- *)
  With[{a = λ Cos[θ], b = λ Sin[θ], c = λ Sin[θ], d = -λ Cos[θ]},
    With[{dvec =  $\begin{pmatrix} x' \\ y' \end{pmatrix}$ , vec =  $\begin{pmatrix} x \\ y \end{pmatrix}$ , mat =  $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ },
      With[(* Valeurs propres et vecteurs propres de la matrice *)
        {esys = Eigensystem[mat], det = Det[mat] },
        With[{
          λ1 = esys[[1, 1]], λ2 = esys[[1, 2]],
          vec1 =  $\begin{pmatrix} \text{esys}[[2, 1]] \\ \text{esys}[[2, 1]] \end{pmatrix}$ , vec2 =  $\begin{pmatrix} \text{esys}[[2, 2]] \\ \text{esys}[[2, 2]] \end{pmatrix}$ 
        },
          (* II. -----Affichages ----- *)
          Grid[{{
            (* A. Texte de droite *)
            Style[Text[Column[{
              Text[TraditionalForm[Row[{"f : ℝ² → ℝ²"}]]],
              Text[TraditionalForm[Row[{"f :  $\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$ "}]]],
              "",
              Text[TraditionalForm[Row[{dvec, " = ", mat, vec}]]],
              Text[TraditionalForm[Row[{"déterminant = ", det}]]],
              "",
              Text[TraditionalForm[Row[{"λ₁ = ", NumberForm[λ1, {5, 2}}]]],
              Text[TraditionalForm[Row[{"λ₂ = ", NumberForm[λ2, {5, 2}}]]],
              "",
              Text[TraditionalForm[Row[{"v₁ = ", NumberForm[vec1, {5, 2}}]]],
              Text[TraditionalForm[Row[{"v₂ = ", NumberForm[vec2, {5, 2}}]]],
              "
              "
            }]], 12],
            (* B. Figure de gauche *)
            VectorPlot[
              {a * x + b * y, c * x + d * y}, {x, -1, 1}, {y, -1, 1},
              StreamPoints → {{p[[1]], p[[2]]}},
              StreamStyle → {Red, Thick}, VectorScale → {Tiny, Automatic, None},
              ImageSize → {250, 250}, FrameLabel → {x, y}
            ]
          }]]],
  ]],
  (* III. -----Contrôles des variables du Manipulate----- *)
  Style["Matrice de Symétrie Orthogonale[λ, θ] :", Bold],
  Control[{{λ, 1, "λ"}, -5, 5, .01, Appearance → "Labeled"}],
  Control[{{θ, 0, "θ"}, -π, π, .01, Appearance → "Labeled"}],
  {{p, {.5, .5}}, {-1, -1}, {1, 1}, Locator}
]

```



## → V. Expressions des Solutions d'EDO avec Mathematica

Nous allons aborder les aspects pratiques et techniques de résolution d'EDO avec *Mathematica* au travers de l'exemple de cinétique de Michaelis-Menten.

Une expression explicite, formelle ou symbolique de la solution d'un système d'équations différentielle est souvent préférable, mais souvent impossible.

Quand cela est possible, *Mathematica* est souvent capable de donner une solution sous forme symbolique. Sinon, il faut recourir à une solution numérique. Examinons les deux cas.

### ■ → A. Résolution Formelle

#### ■ → 1. Échec de résolution formelle d'EDO non linéaires

Pour obtenir la solution formelle d'une EDO avec *Mathematica* 11.0.1 il faudrait utiliser `DSolve[]`.

Mais c'est inutile avec l'EDO ci-dessous, car il n'existe pas de solution sous une forme explicite de ces équations. Bizarrement avec la version 11.0.1, il faut plus de 12 minutes avant de s'en apercevoir....

```
DSolve[{
  (* 2 EDO NON LINEAIRES avec 2 conditions initiales *)
  s'[t] == -(k1 * e0) * s[t] + (k1 * s[t] + kr1) * c[t],
  c'[t] == (k1 * e0) * s[t] - (k1 * s[t] + kr1 + k2) * c[t],
  s[0] == s0,
  c[0] == 0
},
{s[t], c[t]}, t
] // Timing
```

```
{776.041, DSolve[{s'[t] == -e0 k1 s[t] + c[t] (kr1 + k1 s[t]),
  c'[t] == e0 k1 s[t] - c[t] (k2 + kr1 + k1 s[t]), s[0] == s0, c[0] == 0}, {s[t], c[t]}, t]}
```

Essais d'introduction des termes non linéaires : **Reduce[]** (version 11.0.1) et **DSolve[]** (versions 11.0.1, 7.0.5.2, 4.2) ne parviennent pas à donner une solution analytique (ce qui est attendu et normal).

Dans un premier temps, nous allons résoudre explicitement des équations plus simples en ne gardant que les termes linéaires. Remarque : on simplifie beaucoup la solution avec les conditions limites  $s[0]=s_0$ ,  $c[0]=0$ .

→ **1. Résolution formelle d'EDO non linéaires** (solution de l'exercice ici !)

## ■ → 2. Résolution formelle d'EDO linéaires

Solution générale formelle d'un système linéaire : → **2. Résolution formelle d'EDO linéaires** (solution ici !)

```
DSolve[{
  s'[t] == -a1 * s[t] + a2 * c[t],
  c'[t] == a1 - b2 * c[t],
  s[0] == s0, c[0] == 0
}, {s[t], c[t]}, t
] // Simplify
```

$$\left\{ \left\{ c[t] \rightarrow \frac{a1 - a1 e^{-b2 t}}{b2}, \right. \right.$$

$$\left. s[t] \rightarrow \left( -b2 e^{-a1 t} \left( a2 \left( -1 + e^{a1 t} \right) + b2 s0 \right) + a1 \left( a2 - a2 e^{-b2 t} + b2 e^{-a1 t} s0 \right) \right) / \left( (a1 - b2) b2 \right) \right\} \right\}$$

Solution générale formelle d'un autre système linéaire dérivé :

```
DSolve[{
  s'[t] == -a1 * s[t] + a2 * c[t],
  c'[t] == a1 * s[t] + b2 * c[t],
  s[0] == s0, c[0] == 0
}, {s[t], c[t]}, t
] // Simplify
```

$$\left\{ \left\{ c[t] \rightarrow \left( a1 e^{-\frac{1}{2} \left( a1 - b2 + \sqrt{a1^2 + 4 a1 a2 + 2 a1 b2 + b2^2} \right) t} \left( -1 + e^{\sqrt{a1^2 + 4 a1 a2 + 2 a1 b2 + b2^2} t} \right) s0 \right) / \right. \right.$$

$$\left( \sqrt{a1^2 + b2^2 + 2 a1 (2 a2 + b2)} \right),$$

$$\left. s[t] \rightarrow \left( e^{-\frac{1}{2} \left( a1 - b2 + \sqrt{a1^2 + 4 a1 a2 + 2 a1 b2 + b2^2} \right) t} \left( a1 + b2 - a1 e^{\sqrt{a1^2 + 4 a1 a2 + 2 a1 b2 + b2^2} t} - \right. \right. \right.$$

$$\left. b2 e^{\sqrt{a1^2 + 4 a1 a2 + 2 a1 b2 + b2^2} t} + \sqrt{a1^2 + 4 a1 a2 + 2 a1 b2 + b2^2} \left( 1 + e^{\sqrt{a1^2 + 4 a1 a2 + 2 a1 b2 + b2^2} t} \right) \right) s0 \right) / \left( 2 \sqrt{a1^2 + b2^2 + 2 a1 (2 a2 + b2)} \right) \right\} \right\}$$

Champs de vecteurs ...

## ■ B. Résolution numérique

Nous utiliserons les concentrations initiales :  $e_0 = 10^{-3}$  mM,  $s_0 = 1$  mM

Nous allons d'abord résoudre le système d'équations différentielles pour :  $\frac{e_0}{K_m + s_0} \ll 1$

On peut utiliser essentiellement 3 types de méthodes pour introduire les valeurs numériques dans ces résolutions numériques :

- 1/ Programmation Procédurale (Déclarative ou Impérative),
- 2/ Programmation par Règles,
- 3/ Programmation Fonctionnelle.



### ■ 1. Programmation Procédurale (Déclarative ou Impérative) :

Il s'agit de mettre dans la mémoire le plus tôt possible les valeurs numériques indispensables aux calculs. Cette stratégie n'est pas mauvaise ici, puisqu'il s'agit d'une résolution numérique et que la fonction pourrait être compilée avec les valeurs mises en mémoire.

### a./ mettre les valeurs numériques dans l'équation

Cette méthode est immédiate, rapide et directe, mais n'est pas très souple. Un inconvénient important est que les valeurs sont attribuées sans faire explicitement le lien avec des constantes symboliques.

```
solu = NDSolve[
{
  s'[t] == -10 * s[t] + (10 000 * s[t] + 1000) * c[t],
  c'[t] == 10 * s[t] - (10 000 * s[t] + 2000) * c[t],
  s[0] == 1, c[0] == 0
},
{s[t], c[t]},
{t, 0, 10-3}]

{{s[t] → InterpolatingFunction[ Domain: {{0., 0.001}} Output: scalar] [t],
c[t] → InterpolatingFunction[ Domain: {{0., 0.001}} Output: scalar] [t]}}
```

### b./ définir des constantes et fixer leurs valeurs dès le départ

Cette méthode a l'avantage de fixer les valeurs dans la mémoire.

Mais cette méthode a aussi l'inconvénient de fixer les valeurs dans la mémoire.

Or, 95% des erreurs de programmation sont liées à des déclarations dans la mémoire : ce n'est donc pas une très bonne approche sauf si l'on ne veut résoudre qu'un seul problème (et encore !).

```
(k1 = 104; k2 = 103; kr1 = 103; e0 = 10-3; s0 = 1);
solu = NDSolve[
{
  s'[t] == -(k1 * e0) * s[t] + (k1 * s[t] + kr1) * c[t],
  c'[t] == (k1 * e0) * s[t] - (k1 * s[t] + kr1 + k2) * c[t],
  s[0] == s0, c[0] == 0
},
{s[t], c[t]},
{t, 0, 10-3}]
```

## ■ 2/ Programmation par Règles :

Utiliser les règles de substitution, mais il faut alors la faire au moment où l'on pose les équations, car il n'est pas possible de procéder à l'intégration numérique à partir de valeurs symboliques !

Étant donnée la nature du calcul, l'idée est bien la même que précédemment. L'avantage dans cette approche est que les variables restent symboliques et qu'elles n'ont des valeurs numériques que localement.

On distingue donc bien le problème symbolique, et le problème numérique.

```
ClearAll[k1, k2, kr1, e0, s0]
solu = NDSolve[
{
  s'[t] == -(k1 * e0) * s[t] + (k1 * s[t] + kr1) * c[t],
  c'[t] == (k1 * e0) * s[t] - (k1 * s[t] + kr1 + k2) * c[t],
  s[0] == s0, c[0] == 0
} /. {k1 → 104, k2 → 103, kr1 → 103, e0 → 10-3, s0 → 1},
{s[t], c[t]},
{t, 0, 10-3}
]
```

## ■ 3/ Programmation Fonctionnelle (À VALIDER)

Le problème est résolu à l'aide d'une seule fonction. C'est l'approche la plus mathématique qui combine les

avantages de l'approche précédente et la clarté de l'approche procédurale. Il s'agit de poser les valeurs avant d'écrire les équations avec la fonction : **With**[définitions, expression à calculer].

Cette solution est bonne mais elle a l'inconvénient de cacher le symbole `solu`, à l'intérieur du **With**[].

```
With[{k1 = 10^4, k2 = 10^3, kr1 = 10^3, e0 = 10^-3, s0 = 1, tTot = 10^-3},
  solu = NDSolve[
    {
      s'[t] == - (k1 * e0) * s[t] + (k1 * s[t] + kr1) * c[t],
      c'[t] == (k1 * e0) * s[t] - (k1 * s[t] + kr1 + k2) * c[t],
      s[0] == s0, c[0] == 0
    },
    {s[t], c[t]},
    {t, 0, tTot}
  ];
]
```

`solu`

Définition d'une fonction avec des valeurs par défaut :

```
soluFunc[k1_: 10^4, k2_: 10^3, kr1_: 10^3, e0_: 10^-3, s0_: 1, tTot_: 10^-3] := NDSolve[
  {
    s'[t] == - (k1 * e0) * s[t] + (k1 * s[t] + kr1) * c[t],
    c'[t] == (k1 * e0) * s[t] - (k1 * s[t] + kr1 + k2) * c[t],
    p'[t] == k2 * c[t],
    s[0] == s0, c[0] == 0, p[0] == 0
  },
  {s[t], c[t], p[t]},
  {t, 0, tTot}
]

soluFunc[] == soluFunc[(k1*)10^4, (k2*)
  10^3, (kr1*)10^3, (e0*)10^-3, (s0*)1, (tTot*)10^-3];
```

#### ■ 4/ La variable globale "solu"

Dans chacun des cas ci-dessus, en réalité, on récupère la solution à l'aide de la variable globale "solu" qui est une fonction numérique d'interpolation.

`solu` est définie à l'extérieur de la fonction dans les 2 premiers cas, et dans la fonction dans le 3ième cas, mais c'est dans tous les cas une variable globale. Pour que `solu` soit une variable locale, il aurait fallu la mettre dans les définitions entre {} d'un **With**[]. Mais alors nous n'y aurions pas eu accès.

Dans tous les cas, `solu` est une fonction numérique de la variable implicite ou du symbole global `t`.

## VI. Résolution pratique d'EDO numériques

### ■ A. Résolution du système d'EDO pour $\frac{e_0}{K_m + s_0} \ll 1$

On utilise `soluFunc`[] pour définir les équations de type Michaelis-Menten avec les constantes de vitesse suivantes :

$$k_1 = 10^4 \text{ en } \text{mM}^{-1} \text{ s}^{-1}; k_2 = 10^3 \text{ s}^{-1}, kr_1 = 10^3 \text{ s}^{-1}$$

et les concentrations initiales :  $e_0 = 10^{-3} \text{ mM}$ ,  $s_0 = 1 \text{ mM}$  (excès de substrat). On a alors :

$$K_M = \frac{kr_1 + k_2}{k_1} = \frac{10^3 + 10^3}{10^4} = 0.2 \text{ mM}$$

#### ■ 1/ Evaluate : Tracés de la solution (temps court)

On peut tracer les trajectoires solution en faisant, pour `S[t]`. Attention : il n'est toutefois pas possible de tracer la solution en faisant :

```
Plot[(s[t] /. soluFunc[]), {t, 0,  $\frac{1}{10^3}$ },
  AxesLabel → {t, S[t]}, PlotRange → All
]
```

\*\*\* **NDSolve:** 2.042857142857143`\*<sup>-8</sup> cannot be used as a variable.

Mathematica ne peut logiquement pas intégrer cette équation ET la tracer en même temps.

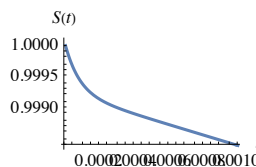
En effet, le même symbole  $t$  est utilisé à la fois :

pour effectuer l'intégration du système d'EDO, et  
pour tenter de faire le tracé.

Il y a deux moyens simples d'éviter ce conflit de symboles.

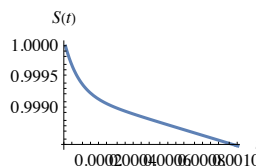
Le premier moyen consiste à récupérer la solution en fonction de  $t$  :  $(s[t]/solu[])$ , puis à remplacer  $t$  dans la solution numérique interpolée par  $tt$ . On trace ensuite cette solution numérique en fonction de  $tt$ .

```
Plot[
  (s[t] /. soluFunc[]) /. t → tt, {tt, 0,  $\frac{1}{10^3}$ },
  AxesLabel → {t, S[t]}, PlotRange → All, ImageSize → 130
]
```

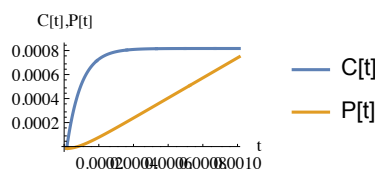


Le moyen recommandé plus simple et plus élégant consiste à utiliser la fonction **Evaluate[]**, qui hiérarchise et dissocie le calcul de la solution numérique et le tracé des solutions :

```
Plot[
  Evaluate[s[t] /. soluFunc[]], {t, 0,  $\frac{1}{10^3}$ },
  AxesLabel → {t, S[t]}, PlotRange → All, ImageSize → 130
]
```



```
Plot[
  Evaluate[{c[t], p[t]} /. soluFunc[]], {t, 0,  $\frac{1}{10^3}$ },
  AxesLabel → {"t", "C[t], P[t]"}, PlotRange → All,
  PlotLegends → {"C[t]", "P[t]"}, ImageSize → 130
]
```



On observe ci-dessus l'établissement dans la première phase, puis le maintien du régime stationnaire à l'échelle de temps très court.

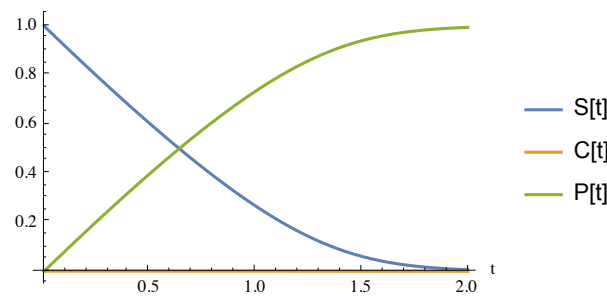
## ■ 2/ Tracés de la solution (temps long 2 secondes = 2000 ms)

Recalculons et mémorisons la solution pour un temps long :

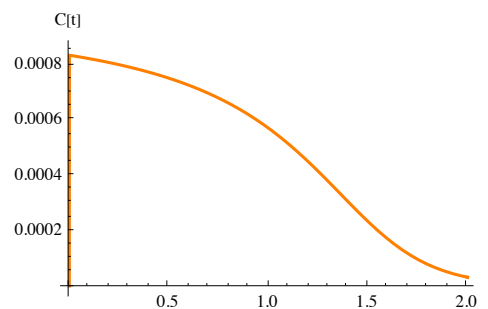
```
ClearAll[solu]
solu = soluFunc[(k1*)10^4, (k2*)
  10^3, (kr1*)10^3, (e0*)10^-3, (s0*)1, (tTot*)2]
```

Traçons les évolutions des concentrations en fonction du temps :

```
Plot[Evaluate[{s[t], c[t], p[t]} /. solu], {t, 0,  $\frac{2000}{10^3}$ },
  AxesLabel -> {"t"}, PlotRange -> All,
  PlotLegends -> {"S[t]", "C[t]", "P[t]"}, ImageSize -> 250]
```



```
Plot[Evaluate[{c[t]} /. solu], {t, 0,  $\frac{2000}{10^3}$ }, PlotStyle -> Orange,
  AxesLabel -> {"t", "C[t]"}, PlotRange -> All, ImageSize -> 250]
```



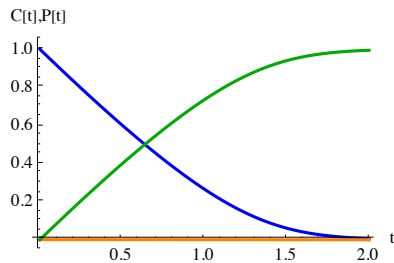
A l'échelle de temps long, on observe ci-dessus la consommation du substrat qui tend vers 0, alors que la formation de produit tend vers 1. L'hypothèse de l'état stationnaire pour le complexe enzyme-substrat n'est admissible que lors de la première phase de la cinétique (la première seconde sans le tout début), comme on peut le voir sur le deuxième graphe.

## ■ 3/ Superposition de graphiques avec Show

Dans l'exemple ci-dessous, on superpose les graphiques gS, gC et gP calculés séparément. Il est ensuite très facile de les tracer ensemble avec **Show[{gS,gC,gP}]** et les options graphiques voulues.

```
gS = Plot[Evaluate[s[t] /. solu], {t, 0,  $\frac{2000}{10^3}$ }, PlotStyle -> Blue];
gC = Plot[Evaluate[c[t] /. solu], {t, 0,  $\frac{2000}{10^3}$ }, PlotStyle -> Orange];
gP = Plot[Evaluate[p[t] /. solu], {t, 0,  $\frac{2000}{10^3}$ }, PlotStyle -> Darker[Green]];
```

```
Show[{gS, gC, gP},
  AxesLabel -> {"t", "C[t], P[t]"},
  PlotRange -> All, ImageSize -> 200
]
```



Ce type de calcul peut être très intéressant dans le cas de nombreux graphiques et de calculs longs par exemples.

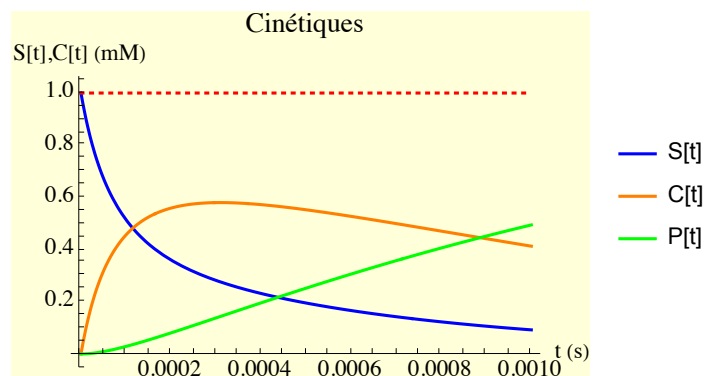
### ■ B. Résolution du système d'EDO pour $\frac{e_0}{K_m + s_0} \approx 1$

On utilise `soluFunc[]` pour définir les équations de type Michaelis-Menten avec les mêmes constantes de vitesse et les concentrations initiales :  $e_0 = 1$  mM,  $s_0 = 1$  mM (épuisement du substrat).

#### ■ 1/ Jolis tracés de la solution

Recalculons et mémorisons la solution pour ce nouveau cas :

```
solu = soluFunc[(k1*)10^4, (k2*)
  10^3, (kr1*)10^3, (e0*)1, (s0*)1, (tTot*)10^-3]
Plot[
  Evaluate[{s[t], c[t], p[t], 1} /. solu, {t, 0, 10^-3}],
  PlotRange -> All, PlotStyle -> {Blue, Orange, Green, {Dotted, Red}},
  PlotLegends -> {"S[t]", "C[t]", "P[t]"},
  AxesLabel -> {"t (s)", "S[t], C[t] (mM)"},
  BaseStyle -> {FontFamily -> "Times", FontSize -> 12, FontColor -> Black},
  Background -> Lighter[Yellow, 0.85],
  PlotLabel -> "Cinétiques", ImageSize -> 300
]
```



#### ■ 2/ Diagrammes de phase en 2D ou 3D

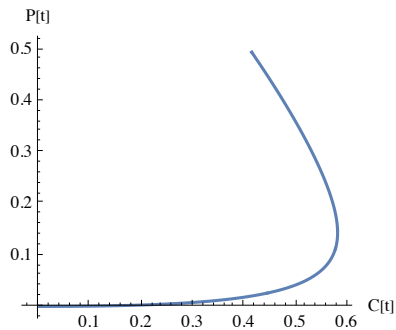
Il est facile de tracer un diagramme de phase en 2D avec `ParametricPlot[]` à partir des solutions calculées :



```

ParametricPlot[
  Evaluate[{c[t], p[t]} /. solu],
  {t, 0, 10-3},
  AxesLabel → {"C[t]", "P[t]"}, PlotRange → All, ImageSize → 200
]

```

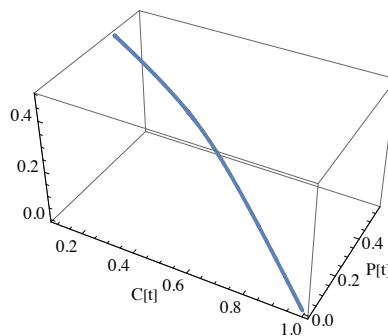


Il est facile de tracer un diagramme de phase en 3D avec ParametricPlot3D[] à partir des solutions calculées :

```

ParametricPlot3D[
  Evaluate[{s[t], c[t], p[t]} /. solu],
  {t, 0, 10-3},
  AxesLabel → {"C[t]", "P[t]"}, PlotRange → All, ImageSize → 200
]

```



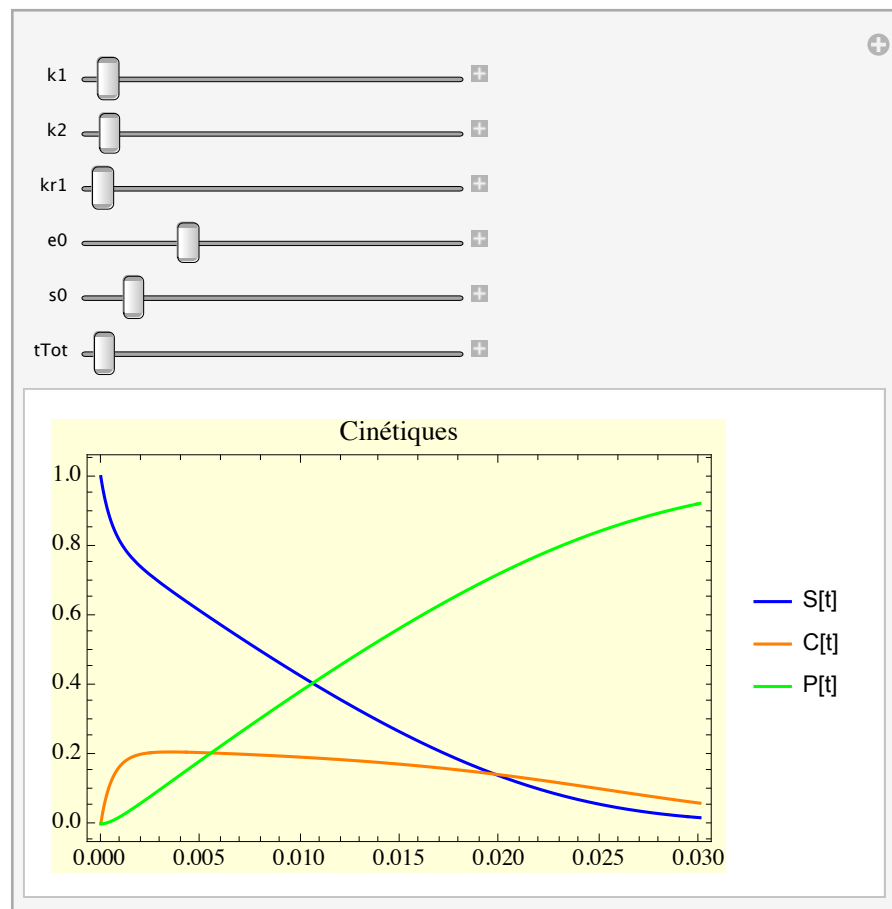
### ■ 3/ Manipulate interactif des solutions (cas général)

Si soluFunc[] est validé, on dispose d'un joli Manipulate efficace et très interactif :

```

ClearAll[a, b, c, d]
Manipulate[
  With[{(* Calcul préliminaire de la solution numérique *)
    solu = soluFunc[k1(*104*), k2
      (*103*), kr1(*103*), e0(*1*), s0(*1*), tTot(*10-3*)]
  },
  Plot[(* Tracés *)
    Evaluate[{s[t], c[t], p[t](*,1*)} /. solu], {t, 0, tTot},
    PlotRange → All, PlotStyle → {Blue, Orange, Green(*,{Dotted,Red}*)},
    PlotLegends → {"S[t]", "C[t]", "P[t]"},
    (*AxesLabel→{"t (s)","S[t],C[t] (mM)"}, Inutile si Frame→True *)
    BaseStyle → {FontFamily → "Times", FontSize → 12, FontColor → Black},
    Background → Lighter[Yellow, 0.85],
    PlotLabel → "Cinétiques", Frame → True, ImageSize → 350
  ]],
  {{k1, 1.5 * 103}, 10, 105},
  {{k2, 0.2 * 103}, 10, 104},
  {{kr1, 0.01 * 103}, 10, 104},
  {{e0 (*mM*), 0.25}, 10-3, 1},
  {{s0 (*mM*), 1}, 0.1, 10},
  {{tTot (*s*), 30 * 10-3}, 10-5, 10}, SaveDefinitions → True
]

```



## Exercices EDO

### → I. Elaboration d'un Manipulate pour EDO

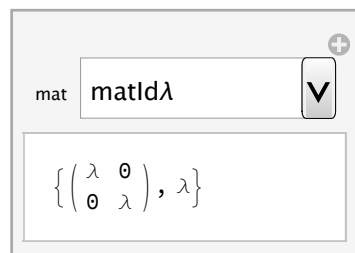
#### ■ → A. VectorPlot de 9 applications linéaires fonctions de $\lambda$ et/ou $\theta$

Champ de vecteurs 9 applications linéaires fondamentales d'angle  $\theta$ , et/ou d'homothétie  $H[\lambda]$  de rapport  $\lambda$

#### ■ → 1. Manipulate pour afficher une matrice parmi 9

En utilisant les définitions de matrices ci-dessous, écrivons un Manipulate pour afficher l'une de ces matrices au choix ainsi que le premier coefficient de la matrice :

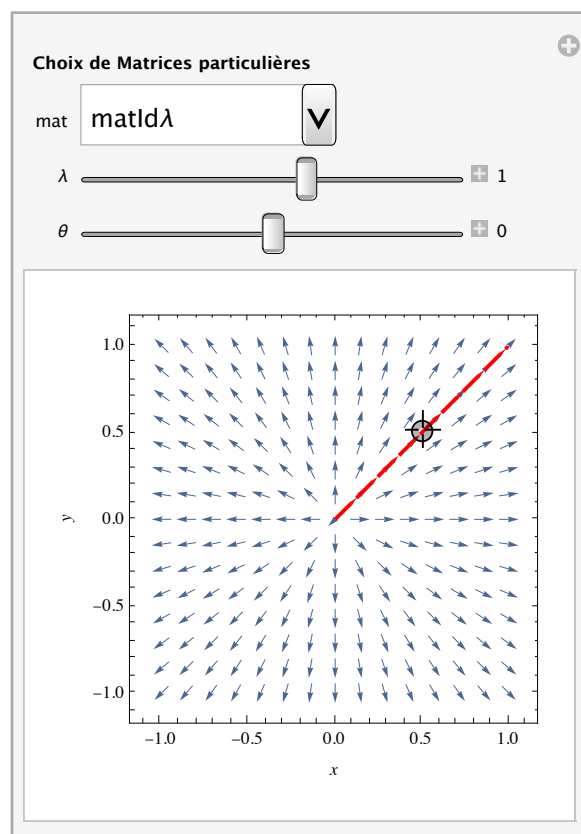
```
(* -----Définition de 9 matrices 2D ----- *)
(* Identite H[λ] *)      matIdλ[λ_, θ_] :=  $\begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix}$ 
(* SymetrieCentrale H[λ] *) matSymCentλ[λ_, θ_] :=  $\begin{pmatrix} -\lambda & 0 \\ 0 & -\lambda \end{pmatrix}$ 
(* Symetrie/X H[λ] *)    matSymXλ[λ_, θ_] :=  $\begin{pmatrix} \lambda & 0 \\ 0 & -\lambda \end{pmatrix}$ 
(* Symetrie/Y H[λ] *)    matSymYλ[λ_, θ_] :=  $\begin{pmatrix} -\lambda & 0 \\ 0 & \lambda \end{pmatrix}$ 
(* Symetrie/Y=X H[λ] *)  matSymYeqXλ[λ_, θ_] :=  $\begin{pmatrix} 0 & \lambda \\ \lambda & 0 \end{pmatrix}$ 
(* SymetrieOrthogonale/axe[θ] H[λ] *) matSymλθ[λ_, θ_] :=  $\begin{pmatrix} \lambda \cos[\theta] & \lambda \sin[\theta] \\ \lambda \sin[\theta] & -\lambda \cos[\theta] \end{pmatrix}$ 
(* Rotation[θ] H[λ] *)   matRotλθ[λ_, θ_] :=  $\begin{pmatrix} \lambda \cos[\theta] & -\lambda \sin[\theta] \\ \lambda \sin[\theta] & \lambda \cos[\theta] \end{pmatrix}$ 
(* Projection[θ] H[λ] *) matProjλθ[λ_, θ_] :=  $\begin{pmatrix} \lambda \cos[\theta]^2 & \lambda \sin[\theta] \cos[\theta] \\ \lambda \sin[\theta] \cos[\theta] & \lambda \sin[\theta]^2 \end{pmatrix}$ 
(* Cisaillement[λ] *)    matCisλ[λ_, θ_] :=  $\begin{pmatrix} 1 & \lambda \\ 0 & 1 \end{pmatrix}$ 
(* ----- *)
(* Manipulate : choix d'une matrice, et affichage du 1er coeff *)
Manipulate[
  {mat[λ, θ] // MatrixForm, mat[λ, θ][[1, 1]]},
  {
    {mat, matIdλ},
    {matIdλ, matSymCentλ, matSymXλ, matSymYλ, matSymYeqXλ,
     matSymλθ, matRotλθ, matProjλθ, matCisλ}}, SaveDefinitions → True
]
```



## ■ → 2. Manipulate pour tracer le champ de vecteurs d'une matrice parmi 9

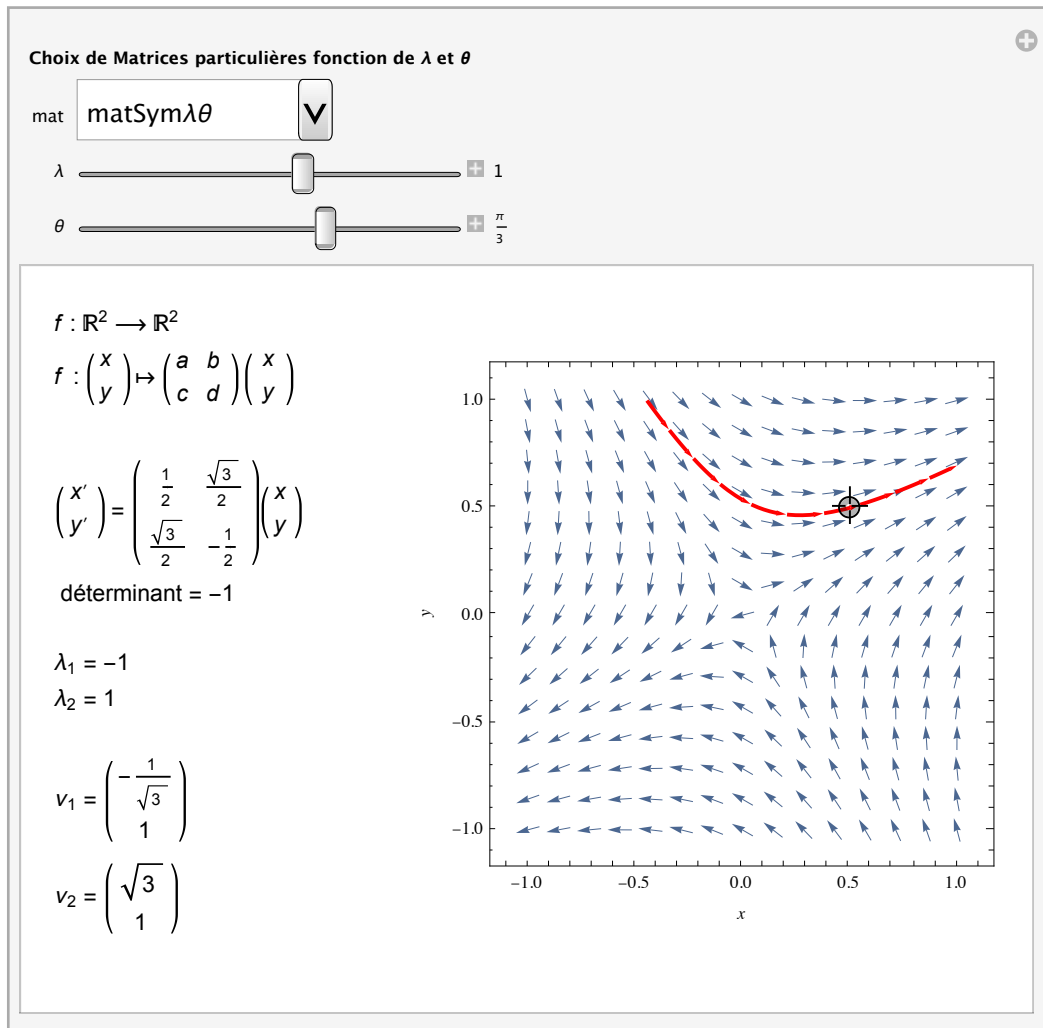
Avec le code vu en préambule, on obtient le Manipulate pour tracer le champ de vecteurs d'une matrice parmi 9 :

```
Manipulate[
  VectorPlot[
    {mat[λ, θ][[1, 1]] x + mat[λ, θ][[1, 2]] y, mat[λ, θ][[2, 1]] x + mat[λ, θ][[2, 2]] y},
    {x, -1, 1}, {y, -1, 1},
    StreamPoints → {{p[[1]], p[[2]]}}, StreamStyle → {Red, Thick},
    VectorScale → {Tiny, Automatic, None},
    ImageSize → {250, 250}, FrameLabel → {x, y}
  ],
  Style["Choix de Matrices particulières ", Bold],
  {{mat, matIdλ}, {matIdλ, matSymCentλ, matSymXλ,
    matSymYλ, matSymYeqXλ, matSymλθ, matRotλθ, matProjλθ, matCisλ}},
  Control[{{λ, 1, "λ"}, -5, 5, .01, Appearance → "Labeled"}],
  Control[{{θ, 0, "θ"}, -π, π, .01, Appearance → "Labeled"}],
  (* Point P des Initialisations de la ligne du champ de vecteurs *)
  {{p, {.5, .5}}, {-1, -1}, {1, 1}, Locator}, SaveDefinitions → True
]
```



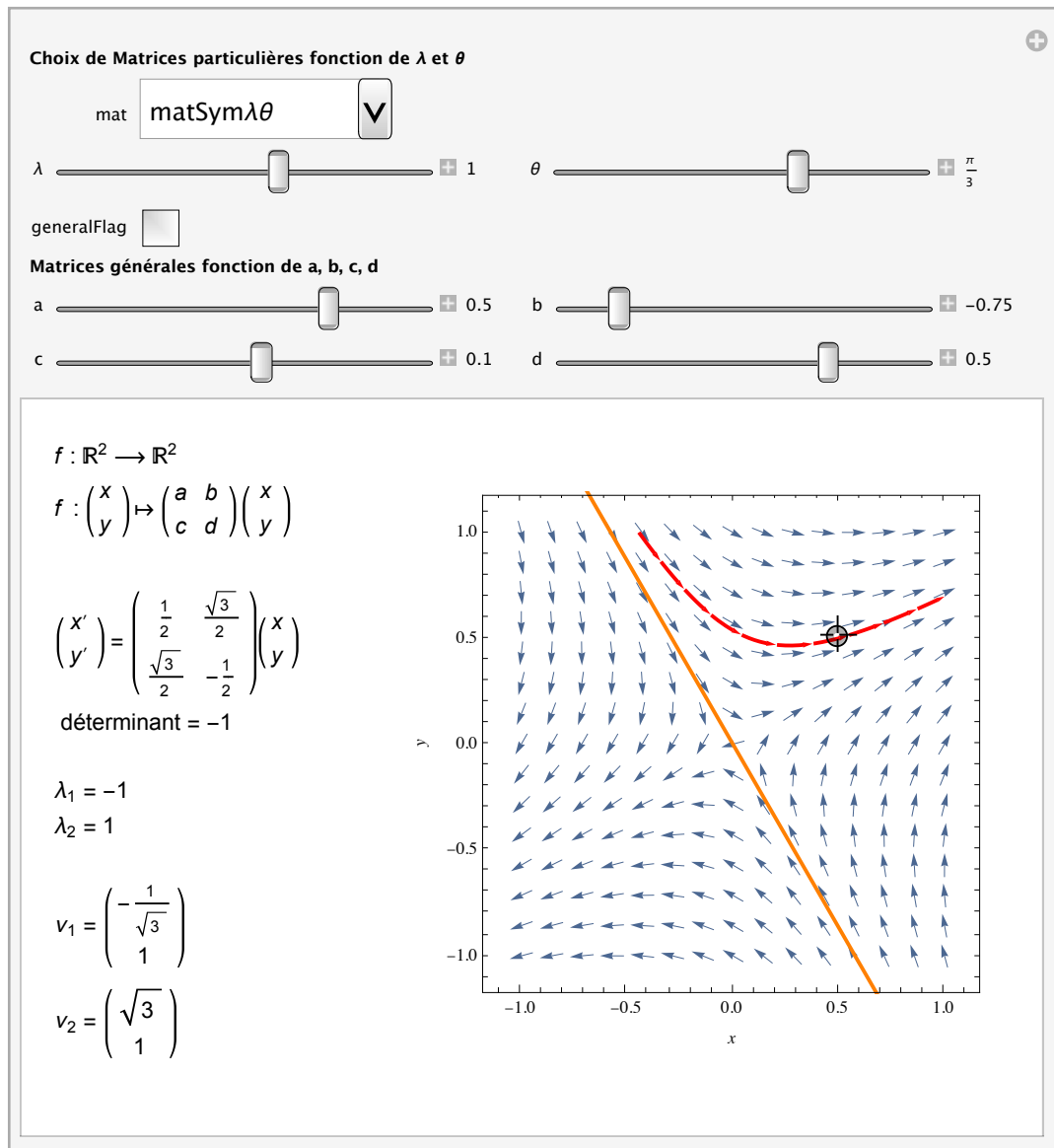
■ → 3. Idem avec Texte et VectorPlot dans deux colonnes adjacentes

Avec le code vu en préambule, on obtient le Manipulate plus complet dont vous étudierez le code et les effets.



■ → 4. Idem + matrice générale + Axes de symétrie et vecteurs propres

On rajoute la possibilité de traiter une matrice réelle générale en activant le “generalFlag”, et en contrôlant les valeurs des coefficients  $a$ ,  $b$ ,  $c$ ,  $d$  de cette matrice.



## ■ → B. VectorPlot de 10 applications linéaires + solutions EDO

### ■ → 1. Idem + calcul et affichage des solutions EDO

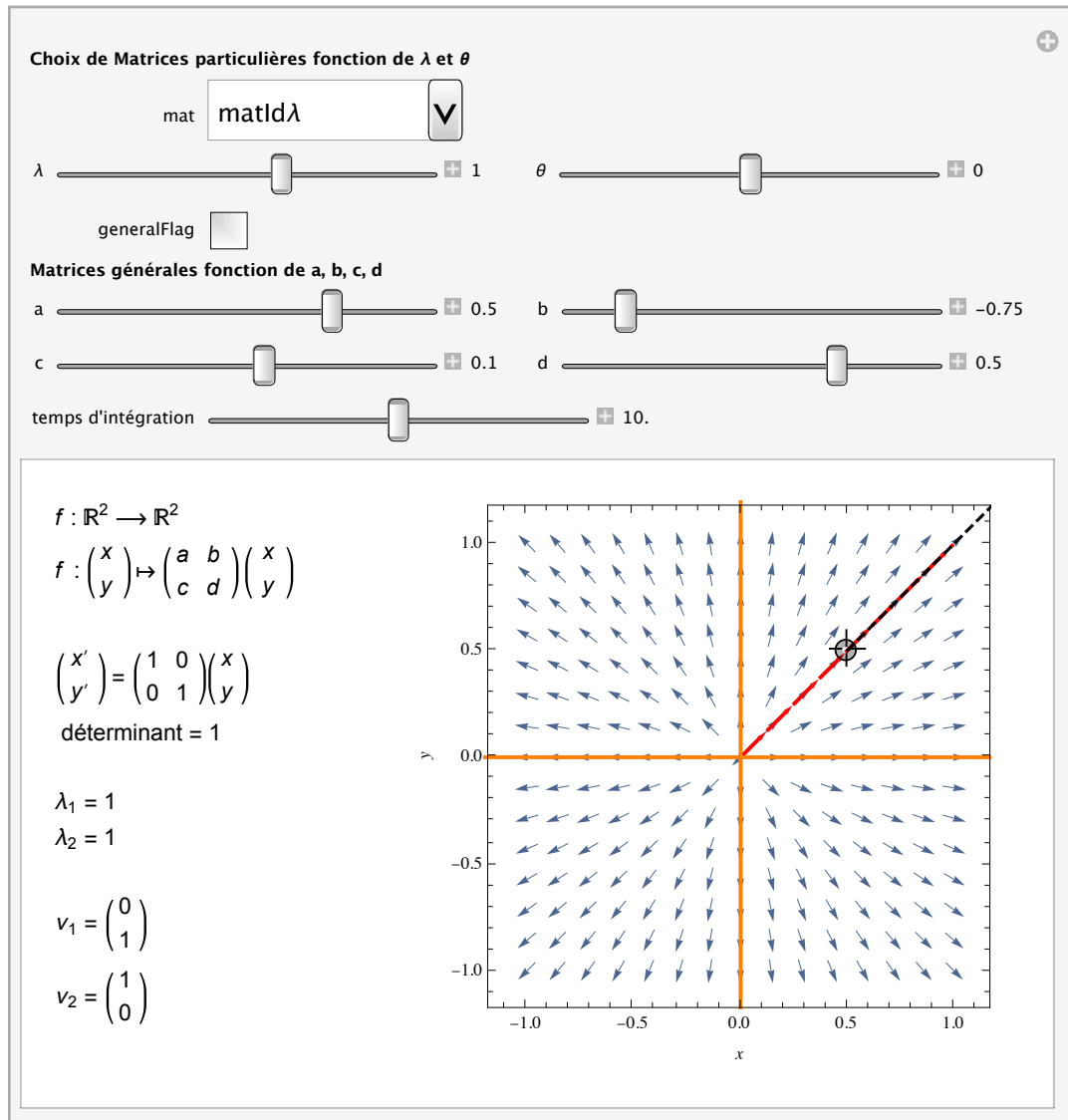
On rajoute :

la partie calcul des EDO à la fin de I. Définitions et calcul des variables, et

le tracé des solutions à la fin de II. Affichage,

le contrôle du temps total de calcul de l'EDO à la fin du III. Contrôles des variables du Manipulate.

La solution de l'EDO est tracée en pointillés noirs à partir du point P donné par le "Locator".



## ■ → C. VectorPlot de 10 (applications + EDO) linéaires + cas non linéaire

### ■ 1. Redéfinition des matrices sans homothétie (À VALIDER)

L'homothétie de rapport  $\lambda$  complique beaucoup les calculs, alors qu'elle n'a pas d'effet dans VectorPlot qui remet à l'échelle. On introduit donc d'abord la simplification des matrices ci-dessous.

Dans le code du Manipulate, il suffit de changer une fois  $\text{mat}[\lambda, \theta]$  en  $\text{mat}[\theta]$  dans le code ci-dessous, et de modifier le contrôle des matrices avec les noms sans  $\lambda$ , mais avec seulement  $\theta$ .

```
(* -----Liste de choix de 9 matrices 2D pour Manipulate ----- *)
(* Simplification en supprimant H[λ] l'homothétie de rapport λ *)
(* Identite *) matId[φ_] :=  $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ 
(* SymetrieCentrale *) matSymCent[φ_] :=  $\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$ 
(* Symetrie/X *) matSymX[φ_] :=  $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ 
(* Symetrie/Y *) matSymY[φ_] :=  $\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$ 
(* Symetrie/Y=X *) matSymYeqX[θ_] :=  $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ 
(* SymetrieOrthogonale/axe[θ] *) matSymθ[θ_] :=  $\begin{pmatrix} \cos[\theta] & \sin[\theta] \\ \sin[\theta] & -\cos[\theta] \end{pmatrix}$ 
(* Rotation[φ] *) matRotθ[θ_] :=  $\begin{pmatrix} \cos[\theta] & -\sin[\theta] \\ \sin[\theta] & \cos[\theta] \end{pmatrix}$ 
(* Projection[φ] *) matProjθ[θ_] :=  $\begin{pmatrix} \cos[\theta]^2 & \sin[\theta] \cos[\theta] \\ \sin[\theta] \cos[\theta] & \sin[\theta]^2 \end{pmatrix}$ 
(* Cisaillement[φ] *) matCisθ[θ_] :=  $\begin{pmatrix} 1 & \theta \\ 0 & 1 \end{pmatrix}$ 
(* ----- *)
```

## ■ → 2. Ajout d'un cas EDO non linéaire (Roméo et Juliette)

Puis on introduit la possibilité de termes non linéaires dans la matrice, et dans l'EDO. Pour cela, il faut procéder aux modifications suivantes.

Dans I. Définitions et calcul des variables, il faut :

\* remplacer la matrice  $\begin{pmatrix} \mathbf{a} & \mathbf{b} \\ \mathbf{c} & \mathbf{d} \end{pmatrix}$  par

$\begin{pmatrix} \mathbf{a} & \mathbf{b} \text{ If}[\text{includeNonlinearTerms}, \text{JM} - \text{Abs}[\mathbf{y}], 1] \\ \mathbf{c} \text{ If}[\text{includeNonlinearTerms}, \text{RM} - \text{Abs}[\mathbf{x}], 1] & \mathbf{d} \end{pmatrix}$

\*\* remplacer `mat[λ,θ]` par `mat[θ]`

\*\*\* faire les substitutions `/.{Abs[y]→ Abs[J[t]]}` et `/.{Abs[x]→ Abs[R[t]]}` dans les EDO.

À la fin du III. Contrôles des variables du Manipulate, il suffit de :

supprimer le contrôle de λ, et

d'ajouter le contrôle des coefficients non linéaires portant sur JM et RM de l'EDO .



Choix de Matrices particulières fonction de  $\lambda$  et  $\theta$ 

mat matId

 $\theta$  0generalFlag ☐

## Matrices générales fonction de a, b, c, d

a 0.5

b -0.75

c 0.1

d 0.5

temps d'intégration 10.

include negative effect of too much love (non linéarité) :

Juliet's love for Romeo  
becomes counterproductive. 0.3Romeo's love for Juliet  
becomes counterproductive. 0.6

$$f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

$$f : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

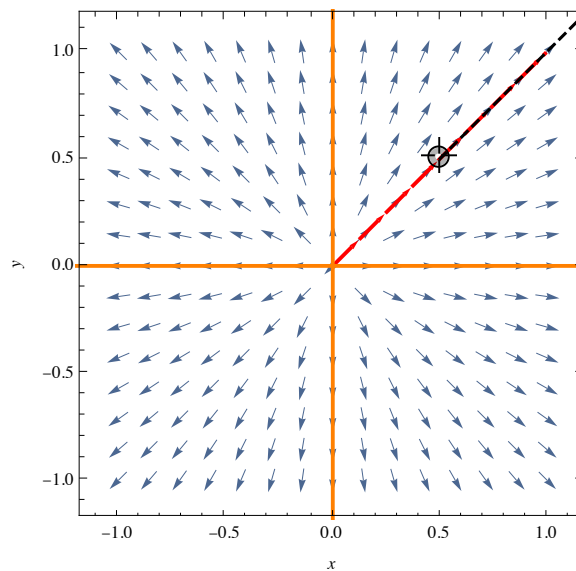
déterminant = 1

$$\lambda_1 = 1$$

$$\lambda_2 = 1$$

$$v_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$v_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$



# Annexes

## I. Caractérisation d'applications linéaires

### ■ A. Étude systématique

#### ■ 1. Présentation

Nous allons étudier systématiquement un certain nombre d'applications linéaires d'usage courant, en donnant :

- 1) la définition de l'application,
- 2) l'image de la transformation au moyen du graphe d'un champ de vecteurs, avec les axes invariants en direction, ou éventuellement, l'absence d'axe. Dans ce graphe, nous avons tracé une droite affine,  $D$ , passant par le point  $A$ , en trait épais, et son image, par la transformation, en trait non épais.
- 3) les images des vecteurs de la base canonique,
- 4) la matrice de l'application linéaire, ou tableau des vecteurs images de la base, et son déterminant,
- 5) une illustration graphique de la transformation,
- 6) le déterminant de la matrice  $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ , noté  $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = \mathbf{Det}\left[\begin{pmatrix} a & b \\ c & d \end{pmatrix}\right] = ad - bc$ .

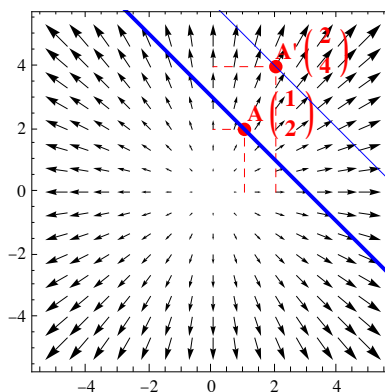
Ces cinq descriptions contiennent essentiellement la même information et sont complémentaires.

### ■ B. Les applications linéaires courantes

#### ■ 1. Homothétie de rapport $\lambda > 0$ et Identité

1) Soit  $f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  avec  $f : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} \lambda x \\ \lambda y \end{pmatrix}$  avec  $\lambda = 2$

2) À cause du facteur d'échelle, l'aspect visuel du champ de vecteurs est identique à celui de l'application identité  $\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \end{pmatrix}$ . Autrement dit,  $\lambda$  ne modifie pas la longueur relative affichée dans le graphe ci-dessous :



On remarque que tous les vecteurs (et les axes passant par  $(0, 0)$ ) sont invariants en direction et en sens.

La droite d'équation paramétrique,  $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 + \lambda \\ 2 - \lambda \end{pmatrix}$ , où  $\lambda \in \mathbb{R}$ , passe par le point  $A \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ , est transformée en la droite parallèle d'équation paramétrique,  $\begin{pmatrix} x \\ y \end{pmatrix} = 2 \begin{pmatrix} 1 + \lambda \\ 2 - \lambda \end{pmatrix}$ . Le point  $A \begin{pmatrix} 1 \\ 2 \end{pmatrix}$  est transformé en  $A' \begin{pmatrix} 2 \\ 4 \end{pmatrix}$ .

3)  $\vec{e}_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix}$  est transformé en  $\lambda \vec{e}_1 \begin{pmatrix} \lambda \\ 0 \end{pmatrix}$ , et  $\vec{e}_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix}$  est transformé en  $\lambda \vec{e}_2 = \begin{pmatrix} 0 \\ \lambda \end{pmatrix}$

4) Matrice :  $\begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \lambda \\ 0 \end{pmatrix} = \lambda \vec{e}_1$ , et  $\begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \lambda \end{pmatrix} = \lambda \vec{e}_2$ ,  $\det \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} = \begin{vmatrix} \lambda & 0 \\ 0 & \lambda \end{vmatrix} = \lambda^2$

La matrice de l'homothétie de rapport  $\lambda$  est  $\lambda$  fois la matrice identité.

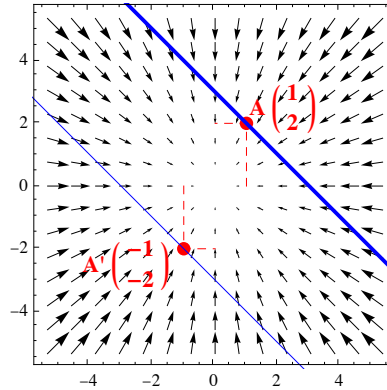
## ■ 2. Symétrie centrale

1) Soit  $f: \mathbb{R}^2 \rightarrow \mathbb{R}^2$

avec

$$f: \begin{pmatrix} x \\ y \end{pmatrix} \mapsto -\begin{pmatrix} x \\ y \end{pmatrix}$$

2) Champ de vecteurs calculé :



En comparant ce graphe avec le précédent 1., on observe que chaque vecteur est inversé (retourné de 180°). Tous les vecteurs sont invariants en direction.

La droite d'équation paramétrique,  $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1+\lambda \\ 2-\lambda \end{pmatrix}$ , où  $\lambda \in \mathbb{R}$ , passe par le point  $A \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ , est transformée en la droite d'équation paramétrique,  $\begin{pmatrix} x \\ y \end{pmatrix} = -\begin{pmatrix} 1+\lambda \\ 2-\lambda \end{pmatrix}$ .  $A \begin{pmatrix} 1 \\ 2 \end{pmatrix}$  est transformé en  $A' \begin{pmatrix} -1 \\ -2 \end{pmatrix}$ .

3)  $\vec{e}_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix}$  est transformé en  $-\vec{e}_1 \begin{pmatrix} -1 \\ 0 \end{pmatrix}$  et  $\vec{e}_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix}$  est transformé en  $-\vec{e}_2 \begin{pmatrix} 0 \\ -1 \end{pmatrix}$

4) Matrice :  $\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix} = -\vec{e}_1$ , et  $\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix} = -\vec{e}_2$ ,  $\det \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{vmatrix} -1 & 0 \\ 0 & -1 \end{vmatrix} = 1$

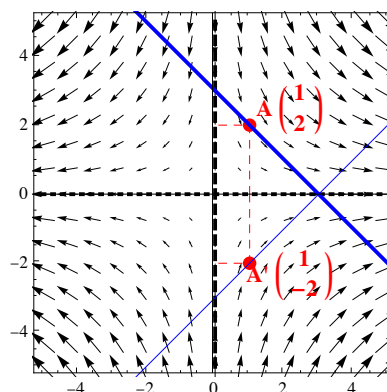
## ■ 3. Symétrie par rapport à l'axe x

1) Soit  $f: \mathbb{R}^2 \rightarrow \mathbb{R}^2$

avec

$$f: \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x \\ -y \end{pmatrix}$$

2) Champ de vecteurs calculé :

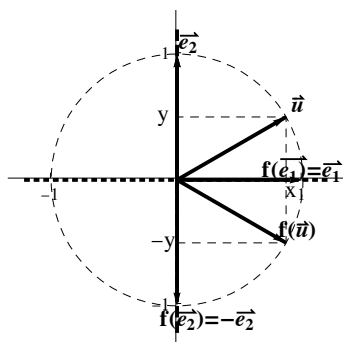


On observe directement que les deux axes, tracés en pointillés, sont invariants en direction. Le premier axe ( $y=0$ ) est aussi invariant en sens (en tirets), tandis que le second ( $x=0$ ) est inversé en sens (en gros points).

3)  $\vec{e}_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix}$  est transformé en  $\vec{e}_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix}$  et  $\vec{e}_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix}$  est transformé en  $-\vec{e}_2 \begin{pmatrix} 0 \\ -1 \end{pmatrix}$

4) Matrice :  $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \vec{e}_1$ , et  $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix} = -\vec{e}_2$   $\det \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{vmatrix} 1 & 0 \\ 0 & -1 \end{vmatrix} = -1$

5)



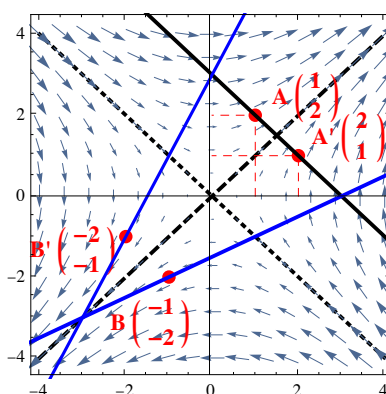
#### ■ 4. Symétrie par rapport à la première bissectrice

1) Soit  $f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$

avec

$$f : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} y \\ x \end{pmatrix}$$

2) Champ de vecteurs calculé :



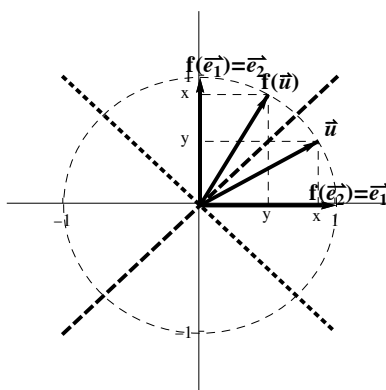
On observe que les deux bissectrices, tracées en pointillés, sont invariantes en direction. La première bissectrice ( $y = x$ ) est aussi invariante en sens (en tirets), tandis que la seconde bissectrice ( $y = -x$ ) est inversée en sens (en gros points).

La droite D est perpendiculaire à la première bissectrice (et parallèle à la seconde bissectrice), est globalement invariante par la transformation. Le point  $A \begin{pmatrix} 1 \\ 2 \end{pmatrix}$  est transformé en  $A' \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ .

La droite d'équation paramétrique,  $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -1 + 2\lambda \\ -2 + \lambda \end{pmatrix}$ , où  $\lambda \in \mathbb{R}$ , passe par le point  $B \begin{pmatrix} -1 \\ -2 \end{pmatrix}$ , est transformée en la droite d'équation paramétrique,  $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} -1 + 2\lambda \\ -2 + \lambda \end{pmatrix}$ .  $B \begin{pmatrix} -1 \\ -2 \end{pmatrix}$  est transformé en  $B' \begin{pmatrix} -2 \\ -1 \end{pmatrix}$ .

3)  $\vec{e}_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix}$  est transformé en  $\vec{e}_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ , et  $\vec{e}_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix}$  est transformé en  $\vec{e}_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

4) Matrice :  $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \vec{e}_2$ , et  $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \vec{e}_1$   $\det \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix} = -1$

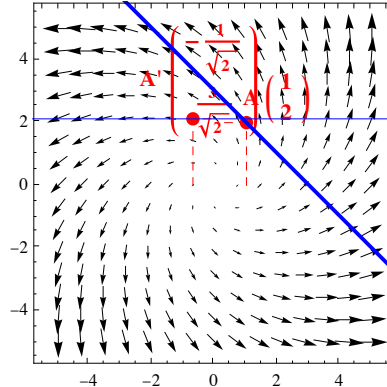


### ■ 5. Rotation d'angle $\theta = 45^\circ$

1) Soit  $f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  avec

$$f : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} \frac{x}{\sqrt{2}} - \frac{y}{\sqrt{2}} \\ \frac{x}{\sqrt{2}} + \frac{y}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \cos[\theta] x - \sin[\theta] y \\ \cos[\theta] x + \sin[\theta] y \end{pmatrix} \text{ où } \theta = \frac{\pi}{4}$$

2) Champ de vecteurs calculé :



On n'observe pas de directions invariantes (réelles) : tous les vecteurs sont tournés de  $45^\circ$ .

La droite d'équation paramétrique,  $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 + \lambda \\ 2 - \lambda \end{pmatrix}$ , où  $\lambda \in \mathbb{R}$ , qui passe par le point  $A \begin{pmatrix} 1 \\ 2 \end{pmatrix}$  est transformée en la droite d'équation paramétrique,  $\begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 + \lambda \\ 2 - \lambda \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 + \lambda - (2 - \lambda) \\ 1 + \lambda + (2 - \lambda) \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 2\lambda - 1 \\ 3 \end{pmatrix}$ .

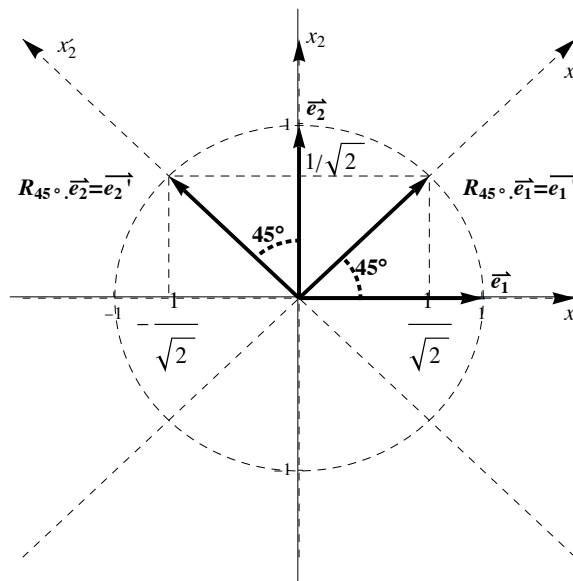
Le point  $A \begin{pmatrix} 1 \\ 2 \end{pmatrix}$  est transformé en  $A' \frac{1}{\sqrt{2}} \begin{pmatrix} 1 - 2 \\ 1 + 2 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 3 \end{pmatrix}$ .

3)  $\vec{e}_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix}$  est transformé en  $f(\vec{e}_1) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$  et  $\vec{e}_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix}$  est transformé en  $f(\vec{e}_2) = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix}$

4) On construit la matrice  $R(45^\circ) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$  à partir du tableau des images, on retrouve

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = R(45^\circ) \vec{e}_1, \text{ et } \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = R(45^\circ) \vec{e}_2, \det \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} = 1$$

5)

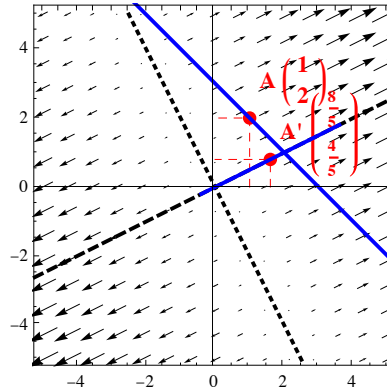


■ 6. Projection sur  $\vec{v} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$

1) Soit  $f : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$  avec

$$f : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \frac{1}{5} \begin{pmatrix} 4 & 2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

2) Champ de vecteurs calculé pour :



On observe qu'il n'y a qu'un seul axe invariant ( $y = 2x$ ), et que l'image de l'axe perpendiculaire est le seul vecteur ( $x = y = 0$ ) figuré par le point ( $x = y = 0$ ).

La droite d'équation paramétrique,  $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 + \lambda \\ 2 - \lambda \end{pmatrix}$ , où  $\lambda \in \mathbb{R}$ , qui passe par le point  $A \begin{pmatrix} 1 \\ 2 \end{pmatrix}$  est transformée en

la droite d'équation paramétrique,  $\begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{5} \begin{pmatrix} 4 & 2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 1 + \lambda \\ 2 - \lambda \end{pmatrix} = \frac{1}{5} \begin{pmatrix} 4(\lambda + 1) + 2(2 - \lambda) \\ 2(\lambda + 1) + 2 - \lambda \end{pmatrix} = \frac{1}{5} \begin{pmatrix} 2(\lambda + 4) \\ \lambda + 4 \end{pmatrix}$ , i.e.

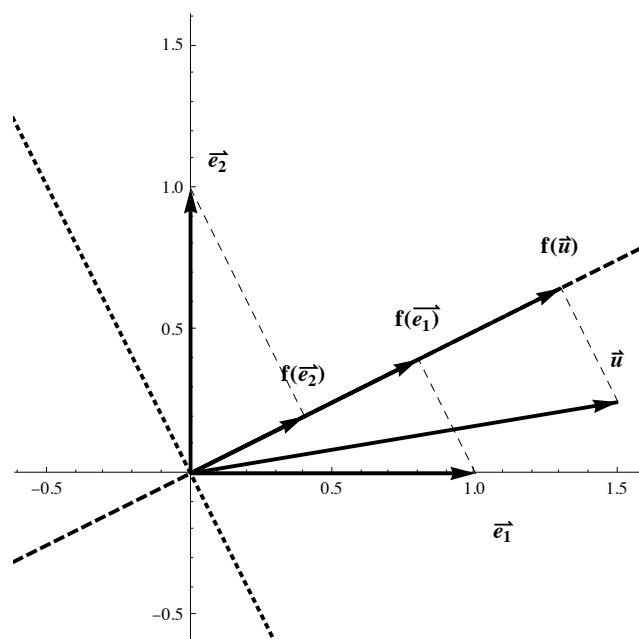
d'équation cartésienne ( $y = 2x$ ) : c'est l'équation de l'axe invariant. En fait on peut montrer que l'image de toute droite du plan est l'axe invariant : c'est une projection sur cette droite.

Le point  $A \begin{pmatrix} 1 \\ 2 \end{pmatrix}$  est transformé en  $A' \frac{1}{5} \begin{pmatrix} 4 & 2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \frac{1}{5} \begin{pmatrix} 8 \\ 4 \end{pmatrix}$ .

3)  $2\vec{e}_1 + \vec{e}_2 \begin{pmatrix} 2 \\ 1 \end{pmatrix}$  est transformé en  $\begin{pmatrix} 2 \\ 1 \end{pmatrix}$ , et  $\vec{e}_1 - 2\vec{e}_2 \begin{pmatrix} 1 \\ -2 \end{pmatrix}$  est transformé en  $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$

4) Matrice :  $\frac{1}{5} \begin{pmatrix} 4 & 2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{5} \begin{pmatrix} 4 \\ 2 \end{pmatrix}$ , et  $\frac{1}{5} \begin{pmatrix} 4 & 2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{5} \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ ,  $\det \begin{pmatrix} \frac{4}{5} & \frac{2}{5} \\ \frac{2}{5} & \frac{1}{5} \end{pmatrix} = \begin{vmatrix} \frac{4}{5} & \frac{2}{5} \\ \frac{2}{5} & \frac{1}{5} \end{vmatrix} = 0$

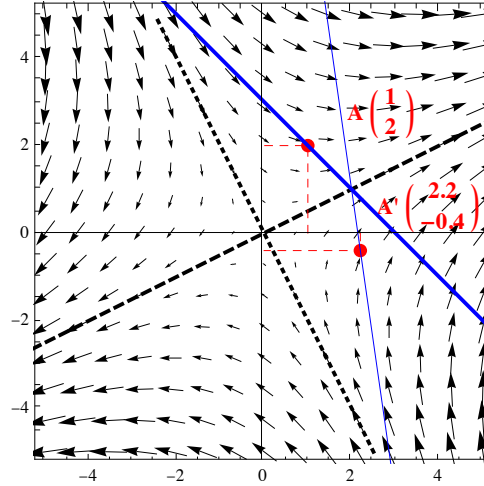
5)



■ 7. Symétrie orthogonale par rapport à  $\vec{v} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$

1) Soit  $f : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$  avec  $f : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \frac{1}{5} \begin{pmatrix} 3 & 4 \\ 4 & -3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

2) Champ de vecteurs calculé pour :



On observe que les deux axes, tracés en pointillés, sont invariants en direction. Le premier axe ( $y = 2x$ ) est aussi invariant en sens (en tirets), tandis que le second ( $y = -\frac{x}{2}$ ) est inversé en sens (en gros points).

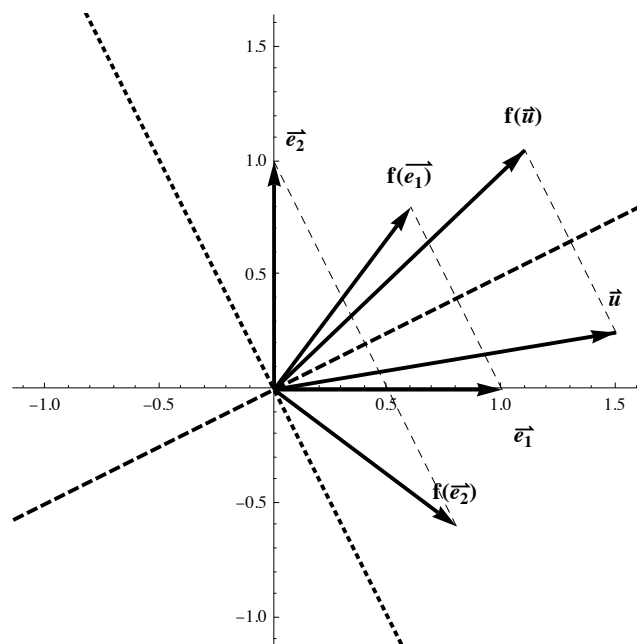
La droite d'équation paramétrique,  $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 + \lambda \\ 2 - \lambda \end{pmatrix}$ , où  $\lambda \in \mathbb{R}$ , qui passe par le point  $A \begin{pmatrix} 1 \\ 2 \end{pmatrix}$  est transformée en

la droite d'équation paramétrique,  $\begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{5} \begin{pmatrix} 3 & 4 \\ 4 & -3 \end{pmatrix} \begin{pmatrix} 1 + \lambda \\ 2 - \lambda \end{pmatrix} = \begin{pmatrix} 2.2 - 0.2\lambda \\ 1.4\lambda - 0.4 \end{pmatrix}$  et  $A \begin{pmatrix} 1 \\ 2 \end{pmatrix}$  en  $A' \begin{pmatrix} 2.2 \\ -0.4 \end{pmatrix}$ .

3)  $2\vec{e}_1 + \vec{e}_2 \begin{pmatrix} 2 \\ 1 \end{pmatrix}$  est transformé en  $\frac{1}{5} \begin{pmatrix} 3 & 4 \\ 4 & -3 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ , et  $\vec{e}_1 - 2\vec{e}_2 \begin{pmatrix} 1 \\ -2 \end{pmatrix}$  en  $\frac{1}{5} \begin{pmatrix} 3 & 4 \\ 4 & -3 \end{pmatrix} \begin{pmatrix} 1 \\ -2 \end{pmatrix} = -\begin{pmatrix} 1 \\ -2 \end{pmatrix}$

4) Matrice :  $\frac{1}{5} \begin{pmatrix} 3 & 4 \\ 4 & -3 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{5} \begin{pmatrix} 3 \\ 4 \end{pmatrix}$ , et  $\frac{1}{5} \begin{pmatrix} 3 & 4 \\ 4 & -3 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{5} \begin{pmatrix} 4 \\ -3 \end{pmatrix}$ ,  $\det \begin{pmatrix} 3/5 & 4/5 \\ 4/5 & -3/5 \end{pmatrix} = \begin{vmatrix} 3/5 & 4/5 \\ 4/5 & -3/5 \end{vmatrix} = -1$

5)

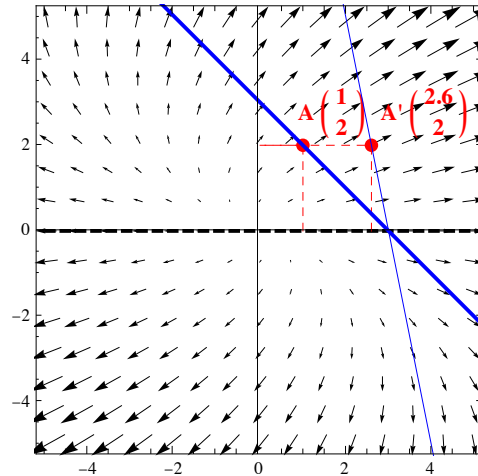


### ■ 8. Cisaillement le long de x

1) Soit  $f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  avec 
$$f : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} 1 & 0.8 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Le cisaillement est d'autant plus fort que l'on s'éloigne de l'axe x.

2) Champ de vecteurs calculé :



On observe directement qu'il n'y a qu'un seul axe invariant ( $y = 0$ ) en direction, et en sens (en tirets).

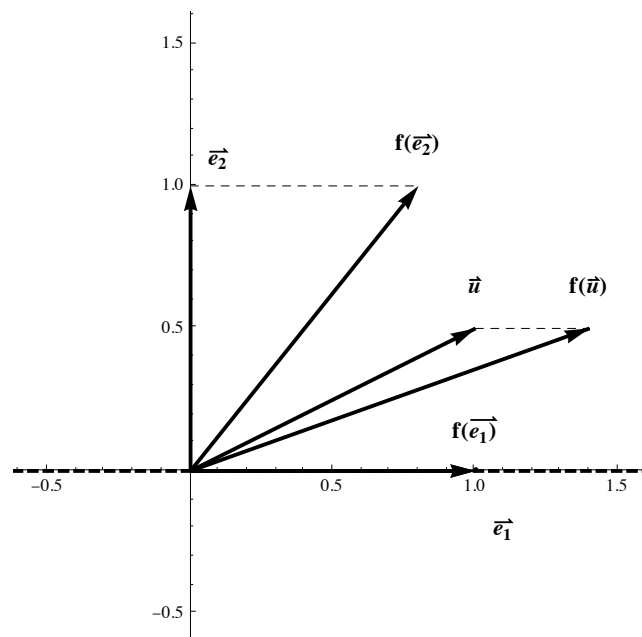
La droite d'équation paramétrique,  $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 + \lambda \\ 2 - \lambda \end{pmatrix}$ , où  $\lambda \in \mathbb{R}$ , qui passe par le point  $A \begin{pmatrix} 1 \\ 2 \end{pmatrix}$  est transformée en

la droite d'équation paramétrique,  $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 0.8 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 + \lambda \\ 2 - \lambda \end{pmatrix} = \begin{pmatrix} 0.2\lambda + 2.6 \\ 2 - \lambda \end{pmatrix}$ .

3)  $\vec{e}_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix}$  est transformé en  $\vec{e}_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ , et  $\vec{e}_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix}$  est transformé en  $0.8 \vec{e}_1 + \vec{e}_2 \begin{pmatrix} 0.8 \\ 1 \end{pmatrix}$

4) Matrice :  $\begin{pmatrix} 1 & 0.8 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ , et  $\begin{pmatrix} 1 & 0.8 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.8 \\ 1 \end{pmatrix}$ ,  $\det \begin{pmatrix} 1 & 0.8 \\ 0 & 1 \end{pmatrix} = \begin{vmatrix} 1 & 0.8 \\ 0 & 1 \end{vmatrix} = 1$

5)





## II. Bibliographie

### ■ A. Références disponibles sur MOODLE 4M062

#### ■ 1. Livres et Articles

<https://www.wolfram.com/language/elementary-introduction/>  
 EIWLStephenWolfram-ElementaryIntroductionWolframLanguage(2015).pdf  
 un-langage-pour-ingenieurThierryVerdel2012.pdf  
 Strogatz-Nonlinear Dynamics and Chaos-00---1994.pdf  
 RomeoJulietSprottNLDPLS2004.pdf

### ■ B. Originaux des 2 Wolfram Demonstrations utilisées

#### ■ PhasePortraitAndFieldDirectionsOfTwoDimensionalLinearSystems

#### ■ HomogeneousLinearSystemOfCoupledDifferentialEquations

## III. Wolfram Demonstrations

Cf. <http://demonstrations.wolfram.com>

Listes non exhaustives classées de Wolfram Demonstrations d'intérêt pour l'UE 4M062, disponibles sur MOODLE.

### ■ A. Les Classiques Indispensables

#### ■ 1BifurcationsCanoniquesLocales

BifurcationDiagramsWithFlowFields-author.nb  
 BifurcationsInFirstOrderODEs-author.nb  
 EffectOfAPerturbationOnTheStablePointsOfADynamicalSystem-author.nb

#### ■ 2ProiesPredateurs

AModelForPopulationGrowth-author.nb  
 PredatorPreyModel-author.nb  
 PredatorPreyEquations-author.nb

#### ■ 3ODE\_Lineaires

CauchysProblem-author.nb  
 FixedPointOfA2x2LinearSystemOfODEs-author.nb  
 HomogeneousLinearSystemOfCoupledDifferentialEquations-author.nb  
 LinearFirstOrderDifferentialEquation-author.nb  
 PhasePortraitAndFieldDirectionsOfTwoDimensionalLinearSystems-author.nb  
 RomeoAndJuliet-author.nb  
 UsingEigenvaluesToSolveAFirstOrderSystemOfTwoCoupledDifferen-author.nb  
 VisualizingTheSolutionOfTwoLinearDifferentialEquations-author.nb

#### ■ 4IntegrationNumerique

EulersMethodForTheExponentialFunction-author.nb

#### ■ 5Attracteurs

LorenzAttractor-author.nb  
 TheRosslerAttractor-author.nb

#### ■ 5Lyapunov

AsymptoticStabilityOfDynamicalSystemByLyapunovsDirectMethod-author.nb  
 FiniteLyapunovExponentForGeneralizedLogisticMapsWithZUnimoda-author.nb  
 LyapunovExponentsForTheLogisticMap-author.nb

#### ■ 5SensibiliteAuxConditionsInitiales

SensitiveDependenceInIteratedMaps-author.nb

SensitiveDependenceOnInitialConditionsInASimpleThreeBodyProb-author.nb  
 SensitivityToInitialConditionsInChaos-author.nb

## ■ B. Suppléments

### ■ 1Bifurcations

BifurcationDiagramForAGeneralizedLogisticMap-author.nb  
 BifurcationDiagramsWithFlowFields-author.nb  
 FeigenbaumsScalingRelationForSuperstableParameterValuesBifur-author.nb  
 HopfBifurcationInTheSelkovModel-author.nb  
 StructuralInstabilityOfASupercriticalPitchforkBifurcation-author.nb  
 TypicalBifurcationsOfWavefrontIntersections-author.nb  
 TypicalBifurcationsOfWavefrontsIn2DAnd3D-author.nb

### ■ 2ProiesPredateurs

ATriangleModelOfCriminality-author.nb  
 BifurcationInAModelOfSpruceBudwormPopulations-author.nb  
 BudwormPopulationDynamics-author.nb  
 ChaosInTumorGrowthModelWithTimeDelayedImmuneResponse-author.nb  
 EcologyOfReefSystems-author.nb  
 EcosystemDynamics-author.nb  
 FoodWebs.cdf  
 PopulationSizesAndInteractionsDuringTheZombieApocalypse-author.nb  
 PredatorPreyDynamicsWithTypeTwoFunctionalResponse-author.nb  
 PredatorPreyEcosystemARealTimeAgentBasedSimulation-author.nb  
 PredatorPreyEquationsSimulatingAnImmuneResponse-author.nb  
 PursuitCurves-author.nb  
 ShellParameterSpace-author.nb  
 SynchronizationOfChaoticAttractors-author.nb

### ■ 3ODE\_NonLineaires

3DVectorFields-author.nb  
 FamiliesOfSolutionsForODEs-author.nb  
 PhasePlanePlotOfTheVanDerPolDifferentialEquation-author.nb  
 TheMurderMysteryMethodForIdentifyingAndSolvingExactDifferent-author.nb

### ■ 4IntegrationNumerique

ASolutionOfEulersTypeForAnExactDifferentialEquation-author.nb  
 ComparingLeapfrogMethodsWithOtherNumericalMethodsForDifferen-author.nb  
 DifferenceEquationVersusDifferentialEquation-author.nb  
 GlobalAndLocalErrorsInRungeKuttaMethods-author.nb  
 NumericalMethodsForDifferentialEquations-author.nb  
 PicardsMethodForOrdinaryDifferentialEquations-author.nb  
 UnderstandingRungeKutta-author.nb

# M3 / Contents

<b>Préambule</b>	1
I. Objectif général et Présentation	1
II. Calcul matriciel	1
A. Définitions des matrices *	1
→ B. Opérations de base sur des matrices *	2
III. Programmation graphique	3
→ A. Opérations graphiques et rotations ... *	3
IV. Champ de Vecteurs	5
A. Introduction	5
B. VectorPlot de l'identité	6
→ C. VectorPlot de Symétrie Orthogonale[ $\lambda, \theta$ ] dans un <del>Manipulate</del>	7
→ V. Expressions des Solutions d'EDO avec Mathematica	10
→ A. Résolution Formelle	10
B. Résolution numérique	11
VI. Résolution pratique d'EDO numériques	13
A. Résolution du système d'EDO pour $\frac{e_0}{K_m + s_0} \ll 1$	13
B. Résolution du système d'EDO pour $\frac{e_0}{K_m + s_0} \approx 1$	16
<b>Exercices EDO</b>	18
→ I. <del>Elaboration d'un Manipulate pour EDO</del>	19
→ A. VectorPlot de 9 applications linéaires fonctions de $\lambda$ et/ou $\theta$	19
→ B. VectorPlot de 10 applications linéaires + solutions EDO	23
→ C. VectorPlot de 10 (applications + EDO) linéaires + cas non linéaire	23
<b>Annexes</b>	26
I. Caractérisation d'applications linéaires	26
A. Étude systématique	26
B. Les applications linéaires courantes	26
II. Bibliographie	32
A. Références disponibles sur MOODLE 4M062	32
B. Originaux des 2 Wolfram <del>Demonstrations</del> utilisées	33
III. <del>Wolfram Demonstrations</del>	33
A. Les Classiques Indispensables	33
B. Suppléments	33