Séance 1 : Modéliser des flux protéiques dans la cellule grâce à l'algèbre linéaire

Cours de L3 : Mathématiques I - ce qu'un biologiste ne peut ignorer

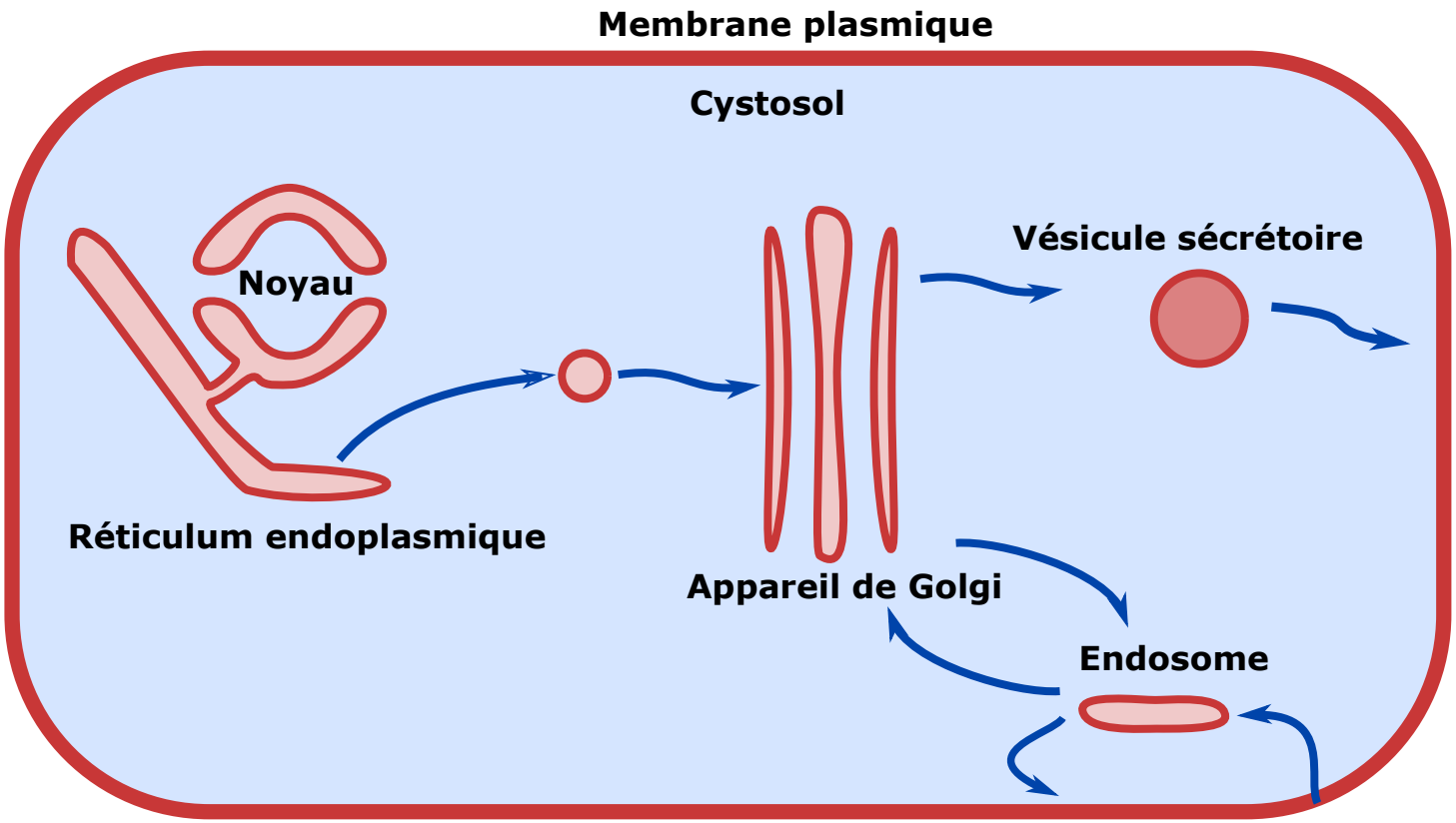
Benoît Perez-Lamarque & Pierre Vincens

ENS Paris - Département de Biologie

13 décembre 2019

Le transport cellulaire des protéines

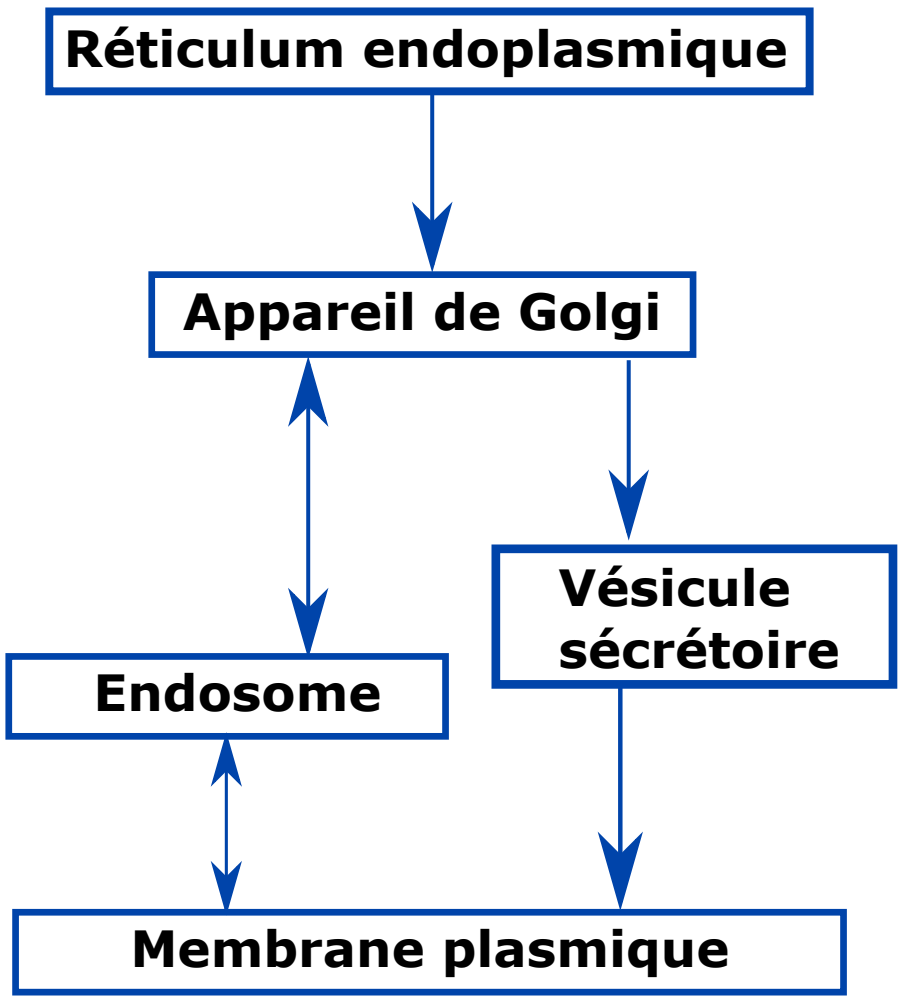
Le transport des protéines dans la cellule eucaryote est un phénomène complexe faisant appel à de nombreux compartiments reliés par des flux de matière très controlés.



Adapté depuis *Biologie moléculaire de la Cellule* (Alberts et. al, 2011) p700

Le transport cellulaire des protéines

Ces flux de matière peuvent être representés sous forme de graphe :



Le transport cellulaire des protéines

On s'intéresse au suivi de protéines fluorescentes marquées à la GFP dans la cellule. On souhaite décrire l'évolution de leur localisation dans les différents compartiments cellulaires, sachant que ces protéines marquées peuvent être **transportées** entre deux compartiments, ou **dégradées**, et que de nouvelles protéines peuvent être **synthétisées**.

Pour cela, nous proposons de réaliser un modèle linéaire en compartiments.

Le transport cellulaire des protéines

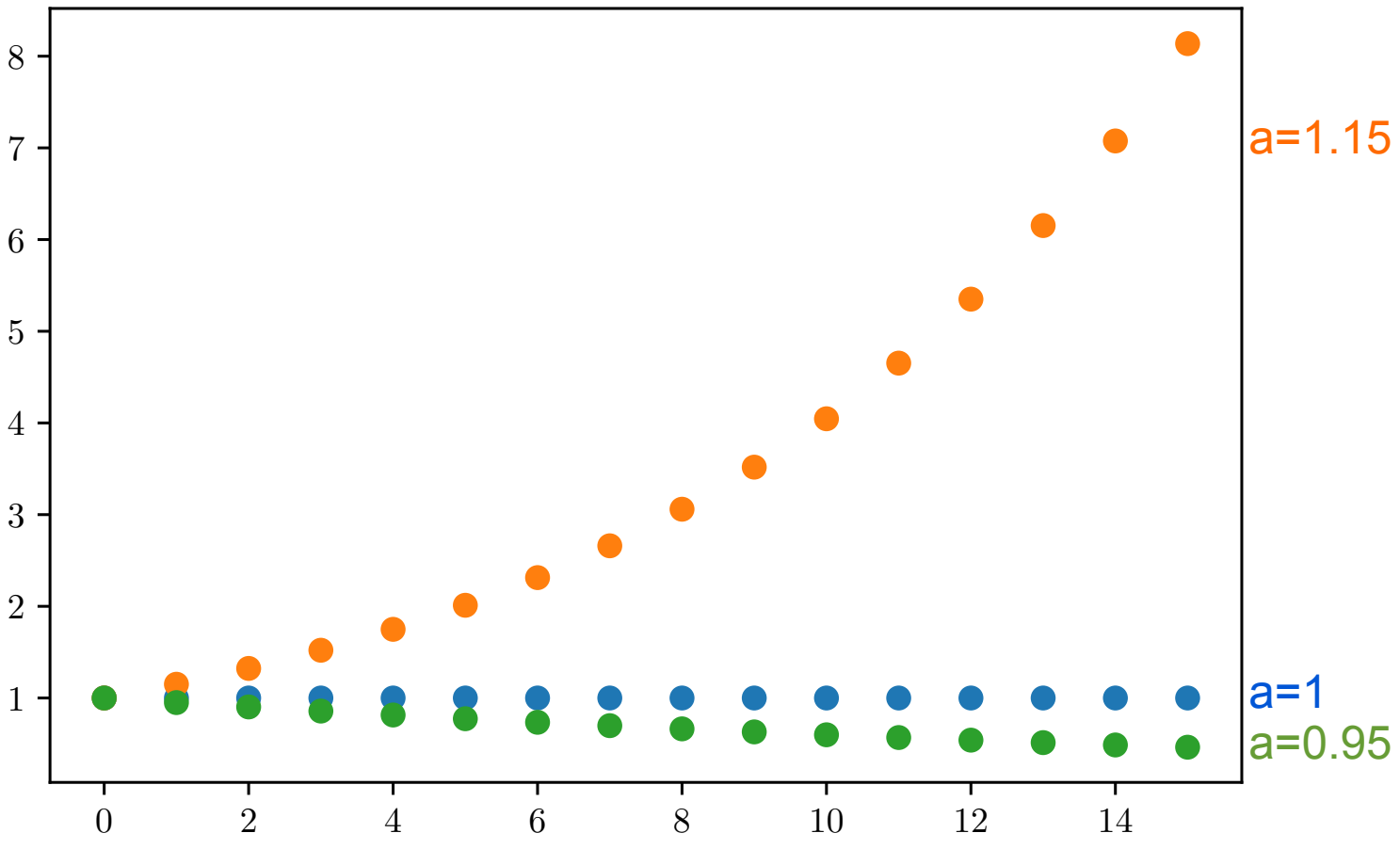
Commençons en une dimension par considérer l'évolution de la quantité totale de protéine dans la cellule, $u(t)$. On suppose qu'à chaque pas de temps, cette quantité est multipliée par une constante a positive (système linéaire - suite géométrique), telle que que

$$u(t+1) = au(t)$$

Exercice 0

Modèle linéaire à 1 dimension

$$u(t+1) = au(t)$$



Modèle linéaire

Pour construire notre modèle à plusieurs compartiments (où **chaque compartiment est représenté par une dimension**), on commence par la réalisation d'un **bilan de matière** pour chaque compartiment dans un intervalle de temps Δ_t :

$$\Delta\text{protéines} = +\text{protéines entrantes} - \text{protéines sortantes}$$

Sachant qu'il y a deux façons d'ajouter ou de retirer des protéines dans un compartiment :

$$\Delta\text{protéines} = \text{flux entrant} + \text{synthèse} - \text{flux sortant} - \text{dégradation}$$

Modèle linéaire

On note $u_{t,i}$ la quantité de protéines dans le compartiment i à l'instant t.
On souhaite donc construire un sytème linéaire de la forme :

$$u_{t+1} = f(u_t)$$

Pour cela, on fait un certain nombre d'hypothèses (cadre du modèle linéaire) :

Modèle linéaire

On note $u_{t,i}$ la quantité de protéines dans le compartiment i à l'instant t.
On souhaite donc construire un sytème linéaire de la forme :

$$u_{t+1} = f(u_t)$$

Pour cela, on fait un certain nombre d'hypothèses (cadre du modèle linéaire) :

- La quantité de protéines transférées d'un compartiment de départ i au compartiment d'arrivée j est proportionnelle à u_i et est indépendante de u_j. Le coefficient de proportionnalité est noté ϕ_{ji} pour les compartiments i de départ et j d'arrivée.
- La quantité de protéines dégradée (resp. synthétisée) dans un compartiment est proportionnelle aux nombre de protéines présentes dans le compartiment. Le coefficient de proportionnalité est noté d_i (resp. s_i) le compartiment i.

Modèle linéaire

Remarques :

- Sachant que la matière est conservée, il faut que $d_i + \sum_j \phi_{ji} \leq 1$
- Ces hypothèses se transcrivent mathématiquement en : $f(u)$ **est une combinaison linéaire des éléments de** u. Ce qui permet d'écrire f comme un produit matriciel. On parle de **modèle linéaire**.

Modèle linéaire

$$(f(u))_j = u_j + s_j u_j - d_j u_j + \sum_{i=1}^{N} \phi_{ji} u_i - \sum_{i=1}^{N} \phi_{ij} u_j \tag{1}$$

$$(f(u))_j = \left(1 + s_j - d_j - \sum_{i=1}^{N} \phi_{ij}\right) u_j + \sum_{i=1}^{N} \phi_{ji} u_i \tag{2}$$

$$(f(u))_j := \sum_{i=1}^{N} A_{ij} u_i \tag{3}$$

$$f(u) = A \circ u \tag{4}$$

Modèle linéaire

Ainsi, la matrice A a pour élément :

$$a_{ij} = \begin{cases} \phi_{ij} & \text{si } i \neq j, \\ 1 + s_i - d_i - \sum_{k=1}^{N} \phi_{ki} & \text{si } i = j. \end{cases} \tag{5}$$

Modèle linéaire

Le modèle que l'on obtient peut s'écrire :

$$\begin{cases} u_t = Au_{t-1} & (\textbf{équation de la dynamique}) \\ u_0 \in \mathbb{R}^N & (\textbf{conditions initiales}) \end{cases} \tag{6}$$

La **suite** $(u_n)_{n\geq 0} \in E^{\mathbb{N}_+}$ est une **trajectoire** du système.

Modèle linéaire

Le modèle que l'on obtient peut s'écrire :

$$\begin{cases} u_t = Au_{t-1} \\ u_0 \in \mathbb{R}^N \end{cases} \tag{7}$$

C'est un modèle :

- **Déterministe** : le comportement du modèle est entièrement déterminée par ses paramètres et ses conditions initiales. Il fait partie de la famille des **Systèmes Dynamiques**.
- **À temps discret**, l'espace du temps est $\mathbb{N}_+$.
- **Autonome**, le temps n'apparaît pas comme variable dans l'équation de la dynamique.
- Dont l'**espace d'état** est $E = \mathbb{R}^N$.
- Dont l'**équation de la dynamique** est une équation linéaire : $u_t = Au_{t-1}$
- Dont les **conditions initiales** sont le vecteur $u_0 \in E$

Modèle linéaire

Le modèle que l'on obtient peut s'écrire :

$$\begin{cases} u_t = A u_{t-1} \\ u_0 \in \mathbb{R}^N \end{cases} \tag{8}$$

Par récurrence, on obtient l'expression analytique du terme générique de la **trajectoire** de ce système :

$$\forall n > 0,\ u_n = A^n u_0$$

La matrice A va ainsi nous permettre de décrire l'évolution du système.

Rappels sur les matrices

Une matrice A de taille $m \times n$ à coefficient dans $\mathbb{C}$ est un ensemble d'éléments de $\mathbb{C}$ ordonnés en m lignes et n colonnes.On note a_{ij} l'élément à la i-ème ligne et k-ème colonne.

$$A = (a_{ij})_{1 \leq i \leq m; 1 \leq j \leq n} \tag{9}$$

On peut aussi commencer l'indexation à 0 pour coller à la représentation informatique :

$$A = (a_{ij})_{0 \leq i < m; 0 \leq j < n} \tag{10}$$

Exemple en taille 3×3 :

$$A = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \tag{11}$$

Rappels sur les matrices

```
# Les matrices peuvent être représentées par des objets de
# type numpy.array.
# Créer un array peut se faire à  partir d'une liste de liste:
A = np.array([[10,6,6],
              [8,2,4],
              [8,4,2]])

# Accéder à  $a_{ij}$ se fait par indexation:
A[0,1] # 6

# Il existe des constructeurs pour des matrices particulières:
I = np.identity(3) # Matrice identité de taille 3
N = np.zeros((3,3)) # Matrice de zeros de taille 3
X,Y = np.meshgrid([1,2,3],[1,2,3]) # Matrices donnant le
# produit cartésien de deux vecteurs (obtention d'un maillage).
```

Rappels sur les matrices

```
# Le produit matriciel pour les np.array peut se faire
# avec la méthode .dot:
x = np.array([1,2,3])
A.dot(x) # Seule possibilité pour Python < 3.5

# Ou l'opérateur @:
A @ x #Python >= 3.5

assert np.all(A.dot(x) ==  A @ x)
```

Rappels sur les matrices

```
# Le produit matriciel n'est défini que pour des matrices de
# dimensions compatibles.
# Exemple, le produit: A * B nécessite que A ait autant de
# colonnes que B ait de lignes.
# (n,m) * (m,k) -> (n,k)

# Par défaut les array numpy de dimension 1 sont orientés à
# chaque opération pour que le produit matriciel fonctionne :
x = np.array([1,2,3])
A@x
x@A

# Pour deux arrays de dimension 1, @ est le produit scalaire.
x@x
```

Rappels sur les matrices

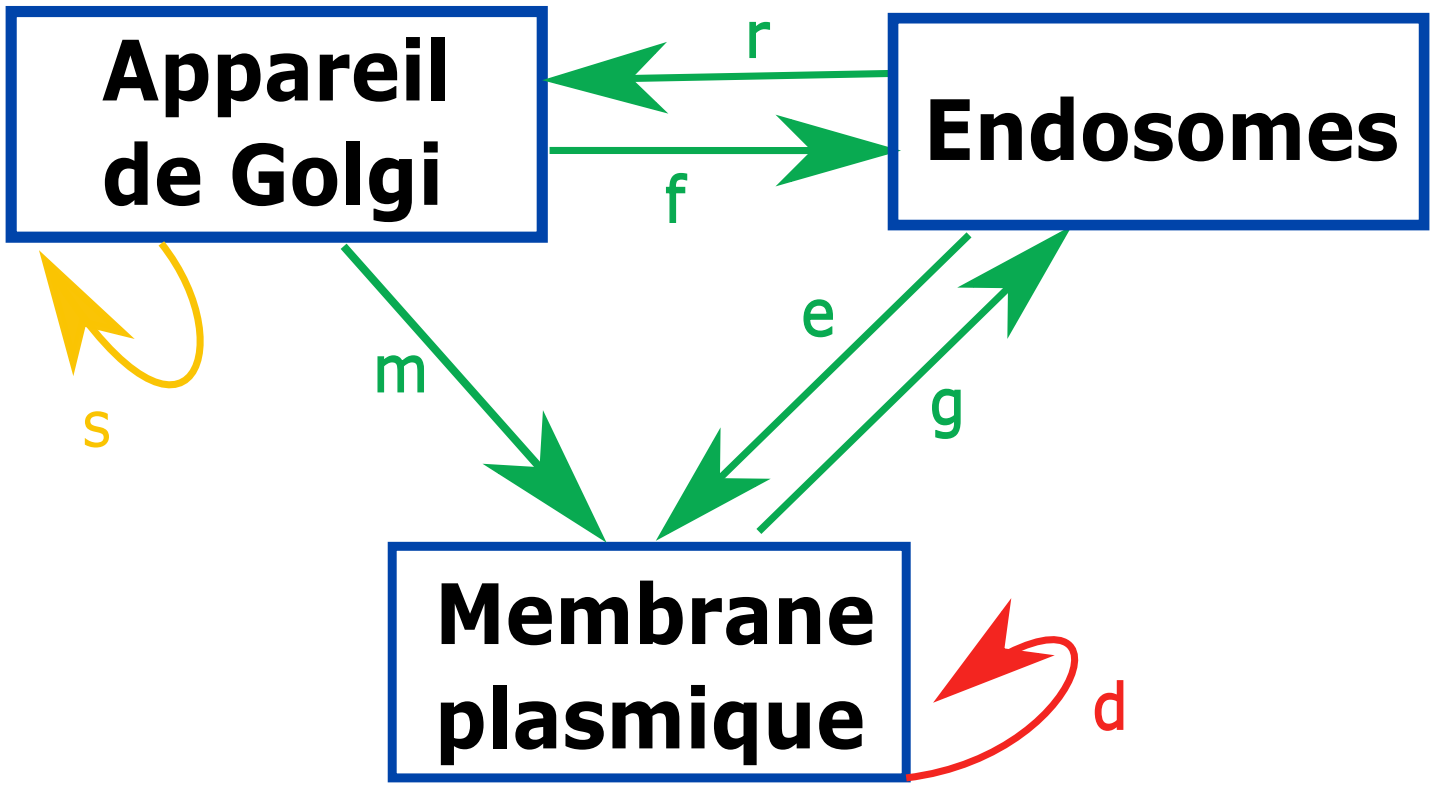
```
# La syntaxe np.newaxis permet d'augmenter sa dimension.
# Maintenant, l'orientation ligne ou colonne est importante.
b = x[:,np.newaxis] # vecteur colonne (3 lignes, 1 colonne)
print("dim:{}, shape:{}, A@b:{}".format(b.ndim, b.shape, A@b))

c = x[np.newaxis,:] # vecteur ligne (1 ligne, 3 colonnes)
print("dim:{}, shape:{}, b@A:{}".format(c.ndim, c.shape, c@A))

c@b # un scalaire (produit scalaire des deux vecteurs <b,c>).
b@c # une matrice 3x3.
```

Partie 1 : Simuler des trajectoires

On suppose une simplification du modèle, où les transports sont représentés en vert, les synthèses en jaune et les dégradations en rouge :



Exercice 1

Partie 1 : Simuler des trajectoires

$$\mathbf{s} = \begin{bmatrix} s \\ 0 \\ 0 \end{bmatrix}, \mathbf{d} = \begin{bmatrix} 0 \\ 0 \\ d \end{bmatrix}, \phi = \begin{bmatrix} 0 & r & 0 \\ f & 0 & g \\ m & e & 0 \end{bmatrix} \tag{12}$$

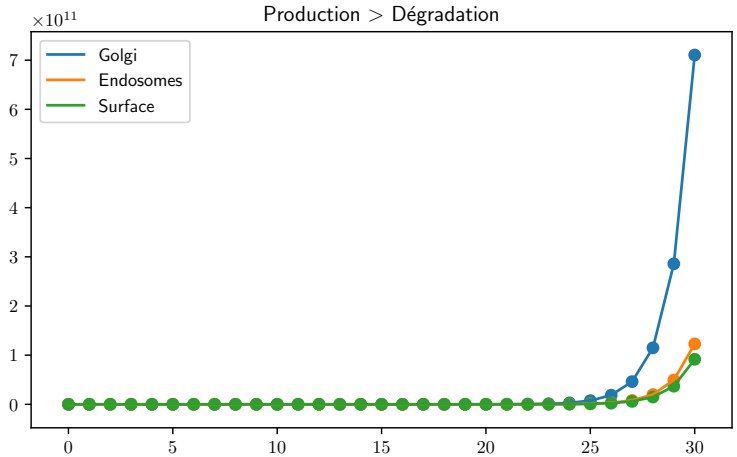
$$\mathbf{A} = \mathbf{I} + \mathbf{Is} - \mathbf{Id} + \phi \tag{13}$$

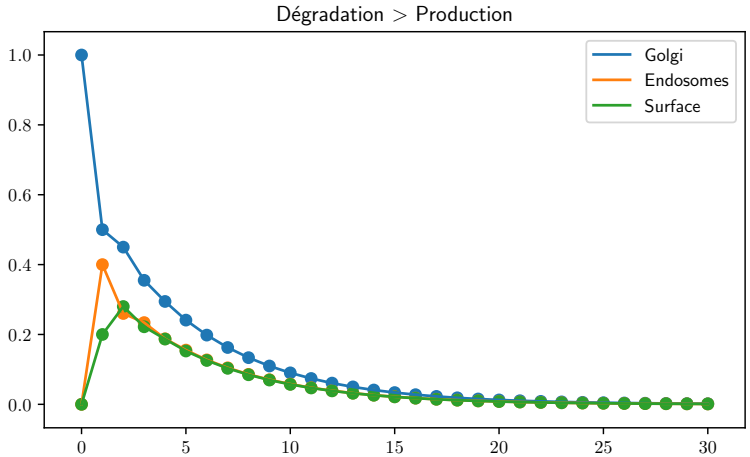
$$\mathbf{A} = \begin{bmatrix} 1 - f - m + s & r & 0 \\ f & 1 - e - r & g \\ m & e & 1 - g - d \end{bmatrix} \tag{14}$$

Exercice 2

Partie 1 : Simuler des trajectoires

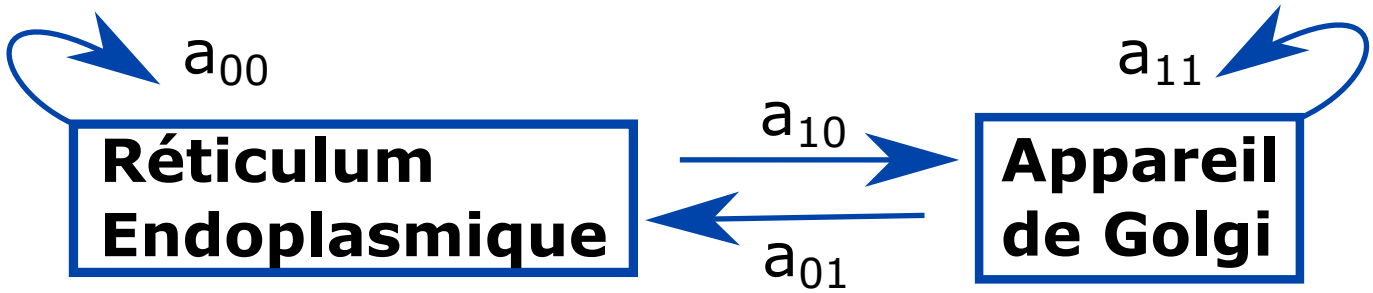
Exercice 2





Partie 2 : matrices 2×2, des endomorphismes sur le plan

Pour encore simplifier et essayer de se construire une intuition géométrique on va passer à des matrices 2×2.



$$A = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} = \begin{bmatrix} 1 + s_0 - d_0 - a_{10} & a_{01} \\ a_{10} & 1 + s_1 - d_1 - a01 \end{bmatrix} \quad (15)$$

Partie 2 : matrices 2×2, des endomorphismes sur le plan

Représenter l'effet d'une matrice :

Une matrice de taille n est une application linéaire de $\mathbb{R}^n$ vers $\mathbb{R}^n$. À tout antécédent $x \in \mathbb{R}^n$ est associé une unique image $Ax \in \mathbb{R}^n$.

Partie 2 : matrices 2×2, des endomorphismes sur le plan

Représenter l'effet d'une matrice :

Une matrice de taille n est une application linéaire de $\mathbb{R}^n$ vers $\mathbb{R}^n$. À tout antécédent $x \in \mathbb{R}^n$ est associé une unique image $Ax \in \mathbb{R}^n$.

On peut représenter l'effet d'une matrice sur un point, sur un nuage de point, mais aussi sur un volume. Dans $\mathbb{R}^2$, on peut regarder l'effet sur tout l'espace en représentant le **champ de vecteur** associé. Une matrice (2×2) est donc une **transformation linéaire du plan**.

Partie 2 : matrices 2×2, des endomorphismes sur le plan

Représenter l'effet d'une matrice :

Une matrice de taille n est une application linéaire de $\mathbb{R}^n$ vers $\mathbb{R}^n$. À tout antécédent $x \in \mathbb{R}^n$ est associé une unique image $Ax \in \mathbb{R}^n$.

On peut représenter l'effet d'une matrice sur un point, sur un nuage de point, mais aussi sur un volume. Dans $\mathbb{R}^2$, on peut regarder l'effet sur tout l'espace en représentant le **champ de vecteur** associé. Une matrice (2×2) est donc une **transformation linéaire du plan**.

L'aire de l'image du carré unité est égal à la **valeur absolue du déterminant**. Le déterminant est négatif si l'orientation du carré unité a changé (les vecteurs de la base canonique se sont croisés).

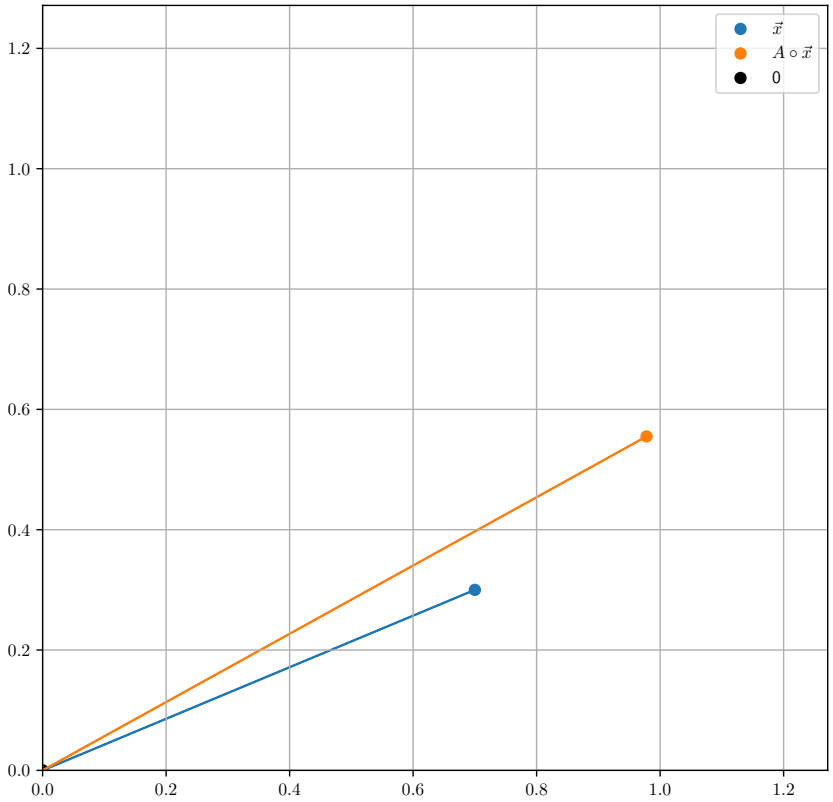
Exercice 3

Partie 2 : matrices 2×2, des endomorphismes sur le plan

Espace de départ	Espace d'arrivée	Représentation graphique	Matplotlib
$\mathbb{R}$	$\mathbb{R}$	courbe, points	'plt.plot', 'plt.scatter'
$\mathbb{R}^2$	$\mathbb{R}$	contour, surface	'plt.contour', 'plt.contourf', 'plt.imshow', 'axes3D.plot_surface'
$\mathbb{R}^2$	$\mathbb{R}^2$	champ de vecteur, écoulement	'plt.quiver', 'plt.streamplot'

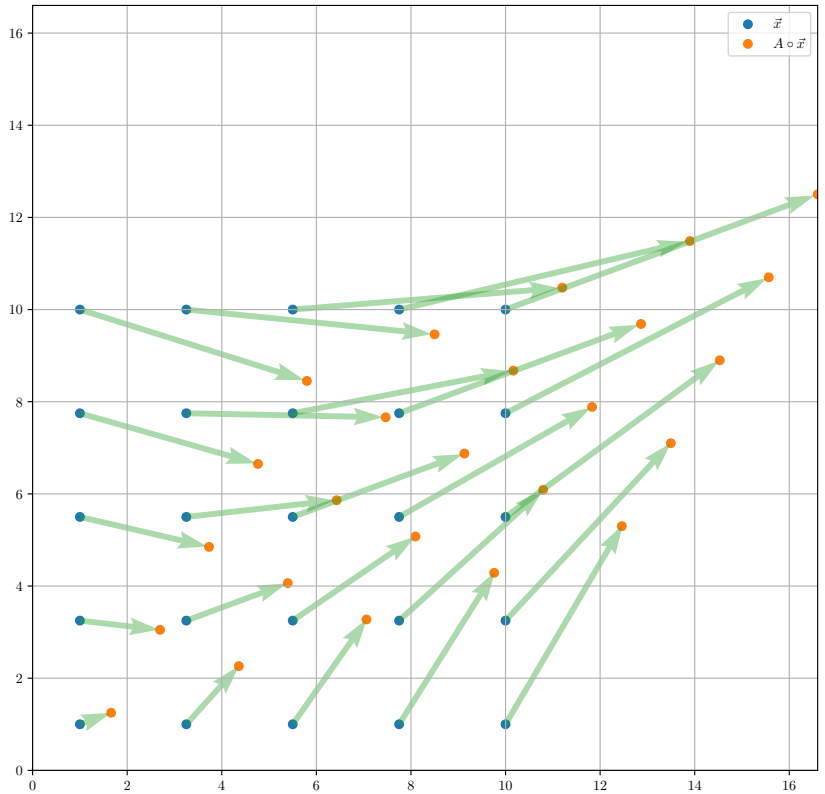
Partie 2 : matrices 2×2, des endomorphismes sur le plan

Représenter une transformation matricielle :



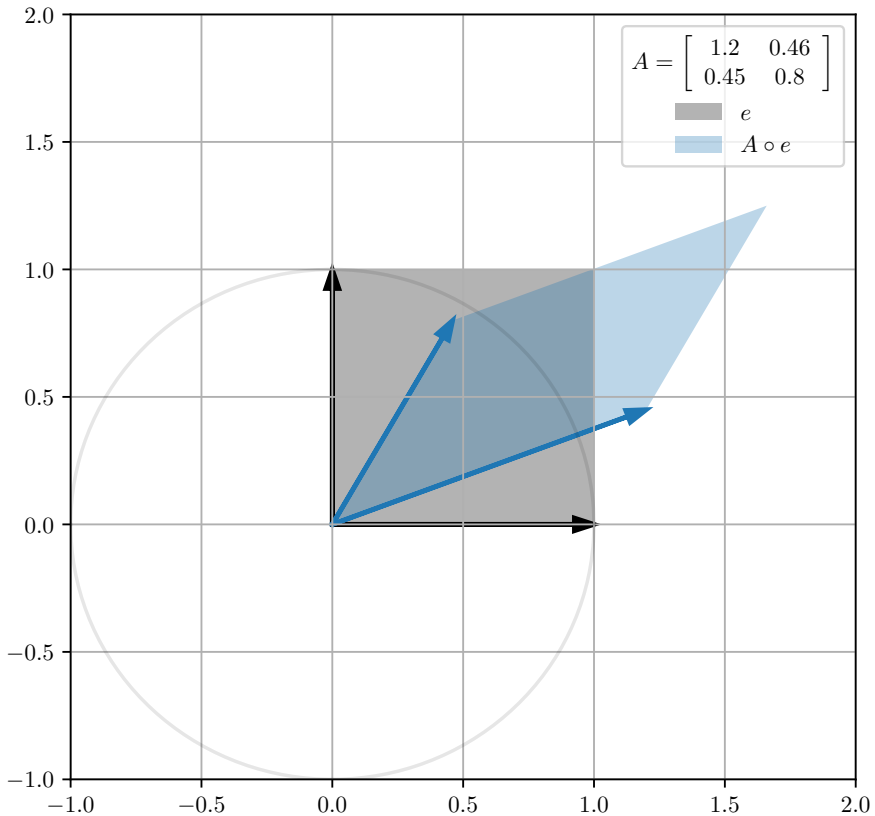
Partie 2 : matrices 2 × 2, des endomorphismes sur le plan

Représenter une transformation matricielle : champs de vecteur



Partie 2 : matrices 2 × 2, des endomorphismes sur le plan

Représenter une transformation matricielle : carré unité



Partie 2 : matrices 2×2, des endomorphismes sur le plan

Produit Matriciel : Composer les transformations :

Multiplier des matrices revient à **composer les transformations**, c'est à dire les appliquer les unes après les autres (en partant de la plus intérieure) : $B \circ C$ revient à appliquer la transformation C, puis B.

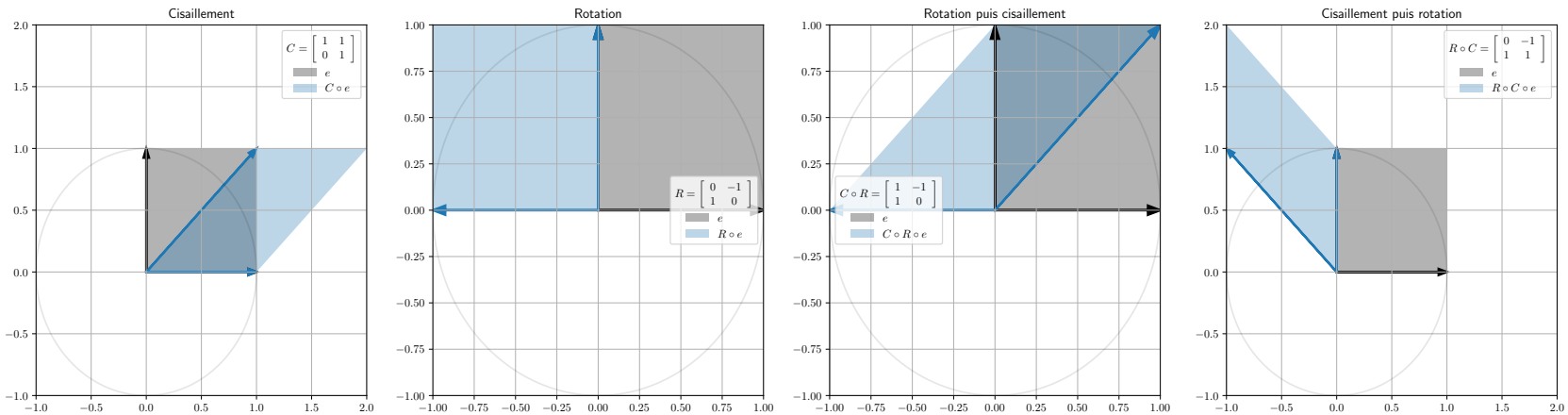
La multiplication de matrice n'est **pas commutative** en général :

$$BC \neq CB$$

Exercice 4

Partie 2 : matrices 2 × 2, des endomorphismes sur le plan

Produit Matriciel : Composer les transformations :



Inverser une matrice : chercher la transformation inverse :

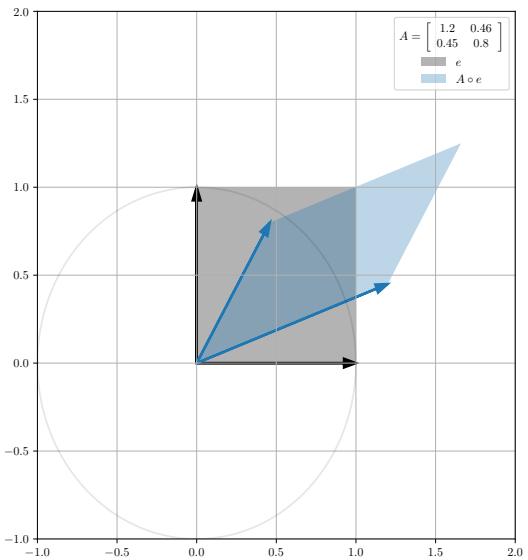
Inverser une matrice revient à trouver la transformation qui annule la transformation associée à cette matrice. Si on garde en tête le fait que la multiplication est une composition des transformations (on en applique l'une puis l'autre) on comprend bien pourquoi :

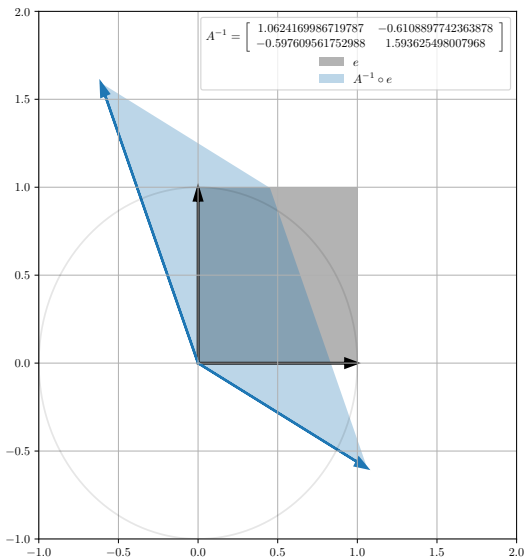
$$A^{-1}A = AA^{-1} = I$$

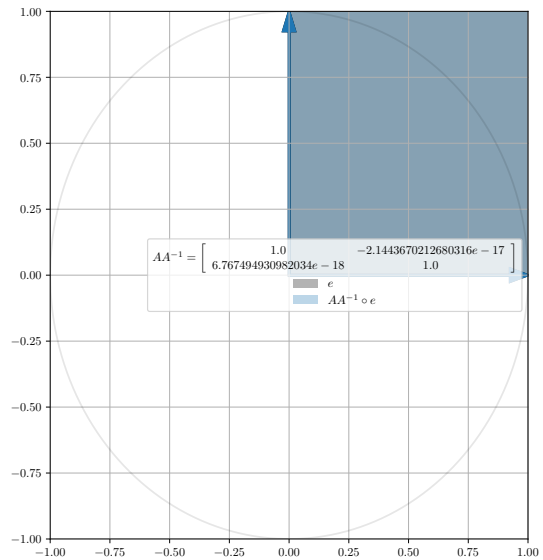
Exercice 5

Partie 2 : matrices 2×2, des endomorphismes sur le plan

Inverser une matrice : chercher la transformation inverse :

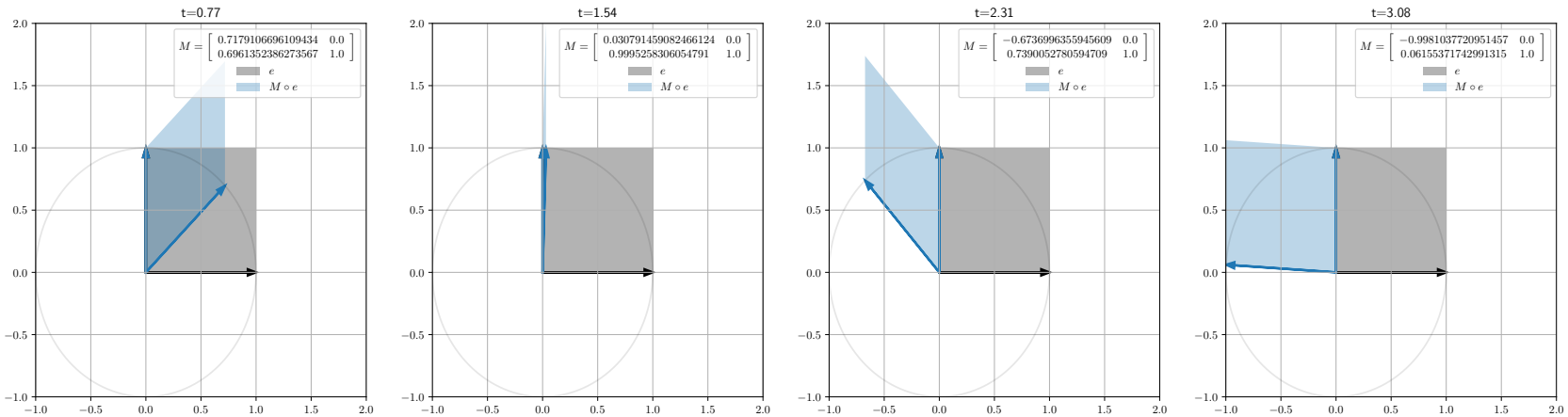






Partie 2 : matrices 2×2, des endomorphismes sur le plan

Inverser une matrice : matrices non inversibles :



Partie 2 : matrices 2×2, des endomorphismes sur le plan

Diagonaliser une matrice : Chercher les directions pour lesquelles la transformation est une dilatation :

Une matrice est **diagonalisable** si la transformation associée est une dilatation non homogène dans une certaine base.

Par définition, les vecteurs propres d'une matrice sont des vecteurs u pour lesquels :

$$Au = \lambda u$$

où λ est la valeur propre associée au vecteur propre u.

Partie 2 : matrices 2×2, des endomorphismes sur le plan

Diagonaliser une matrice A revient à trouver P (inversible) et D (diagonale) telles que :

$$A = PDP^{-1}$$

où les colonnes de P sont les vecteurs propres de A et les valeurs de la diagonale de D sont les valeurs propres associées.

Partie 2 : matrices 2×2, des endomorphismes sur le plan

Diagonaliser une matrice A revient à trouver P (inversible) et D (diagonale) telles que :

$$A = PDP^{-1}$$

où les colonnes de P sont les vecteurs propres de A et les valeurs de la diagonale de D sont les valeurs propres associées.

Cela signifie que la transformation A est équivalente à :

- **i) Transformer le plan** tel que les vecteurs exprimés dans la base canonique soient exprimés dans la base formée par les vecteurs propres (c'est à dire appliquer P^{-1}).
- **ii) Réaliser la dilatation non homogène** : agrandir ou réduire les vecteurs de la base en les multipliant par un scalaire (c'est à dire appliquer D).
- **iii) Re-transformer le plan** pour revenir à la base canonique (en appliquant P).

Partie 2 : matrices 2×2, des endomorphismes sur le plan

Diagonaliser une matrice A revient à trouver P (inversible) et D (diagonale) telles que :

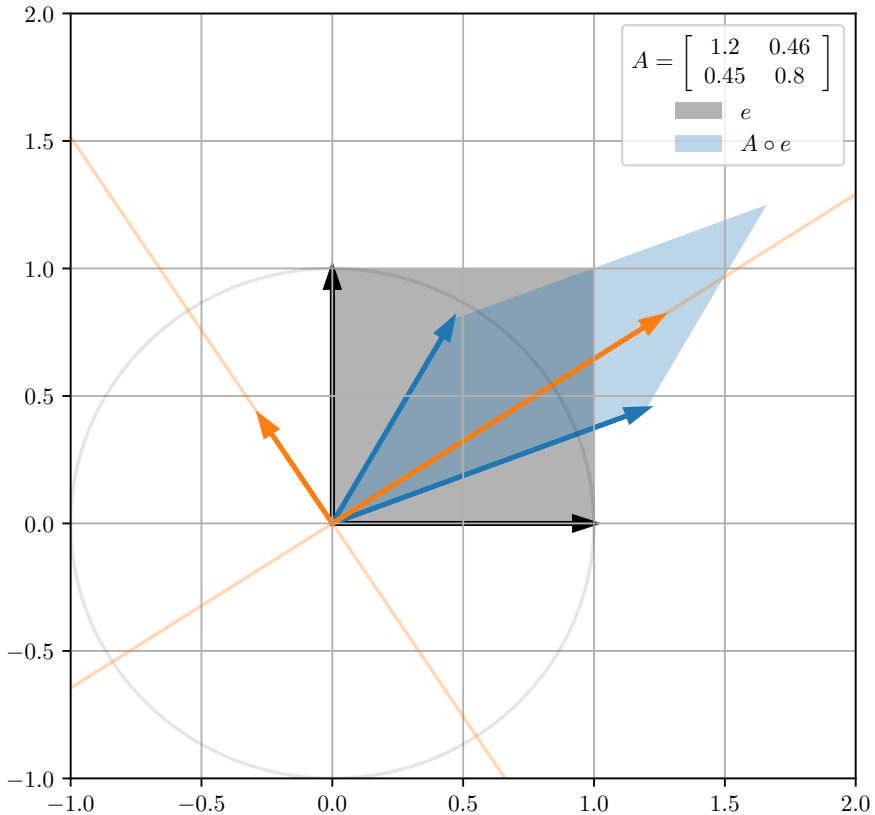
$$A = PDP^{-1}$$

où les colonnes de P sont les vecteurs propres de A et les valeurs de la diagonale de D sont les valeurs propres associées.

Exercice 6

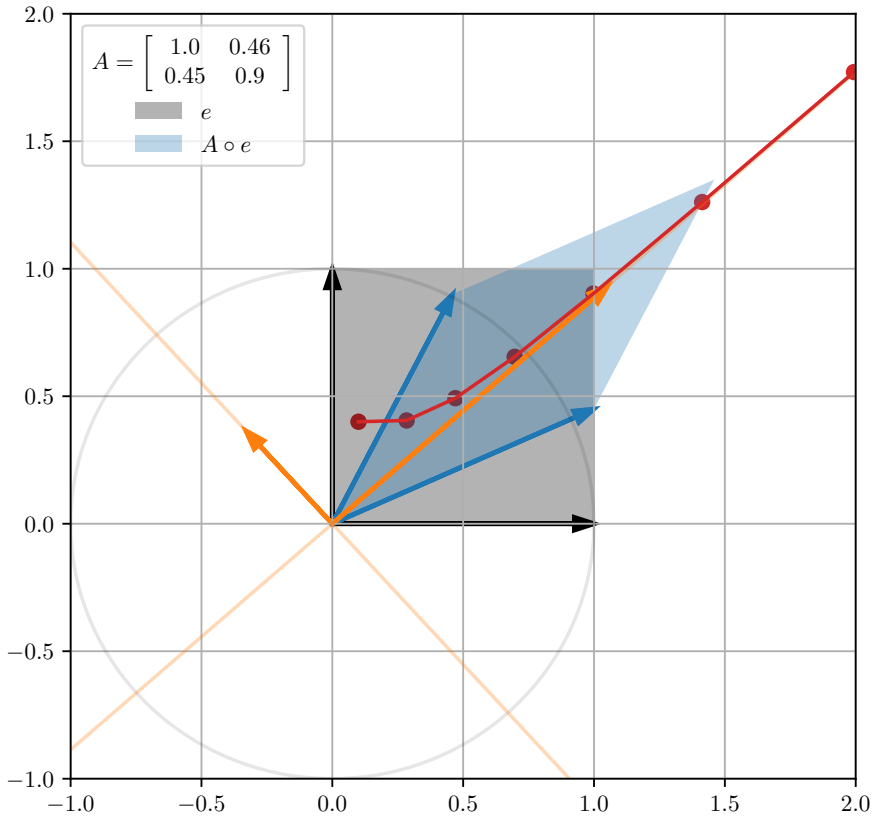
Partie 2 : matrices 2×2, des endomorphismes sur le plan

Les axes propres représentent les axes de dilation de la transformation :



Partie 2 : matrices 2×2, des endomorphismes sur le plan

Les axes propres représentent les axes de dilation de la transformation et permettent de directement prédire la trajectoire du système.



Partie 2 : matrices 2×2, des endomorphismes sur le plan

Le théorème de Perron-Frobenius donne les conditions pour lesquelles une matrice A a une **valeur propre dominante** :

Définition : On dit que la matrice A est positive (noté $A \geq 0$) au sens de Perron-Frobenius si $\forall i, j,\ A_{ij} \geq 0$.

Définition : On dit que la matrice A est régulière s'il existe un entier n pour lequel $A^n > 0$.

Définition : On dit que la matrice A est irréductible s'il existe un entier n pour lequel $\forall i, j,\ (A^n)_{ij} \geq 0$.

Partie 2 : matrices 2×2, des endomorphismes sur le plan

Le théorème de Perron-Frobenius donne les conditions pour lesquelles une matrice A a une **valeur propre dominante** :

Théorème de Perron-Frobenius :

1) Soit $A \geq 0$, alors il existe une valeur propre λ_0 telle que $\forall \lambda \in sp(A),\ |\lambda_0| > |\lambda|$.

2) Soit $A \geq 0$ et régulière, alors λ_0 est de multiplicité 1.

3) Soit A une matrice positive et irréductible, alors (i) A possède une plus grande valeur propre λ_0, (ii) λ_0 est réel et de multiplicité 1, et (iii) le vecteur propre associée à λ_0 est réel et à coefficient positifs.

Exercice 7