MTH 8302 - Modèles de Régression et d'Analyse de Variance

Leçon 0: Rappel d'Algèbre Linéaire, d'Optimisation, de Probabilités, de Statistique et de Python

Polytechnique Montréal - Hiver 2025

Chiheb Trabelsi

January 15, 2025

Outline

Table des Matières

Constantes, Vecteurs et Matrices

Constantes, Vecteurs et Matrices

- **Les constantes** (représentées par des lettres minuscules en italique)
  - Notation : a, b, c.
  - Dimension : $a \in \mathbb{R}$, un nombre réel.
- **Les vecteurs** (représentés par des lettres minuscules en gras)
  - Notation: $\mathbf{v}, \mathbf{x}, \mathbf{y}$.
  - Dimension : $\mathbf{v} \in \mathbb{R}^n$, un vecteur réel de dimension n.
- **Les matrices** (représentées par des lettres majuscules en gras)
  - Notation : $\mathbf{A}, \mathbf{X}$
  - Dimension : $\mathbf{A} \in \mathbb{R}^{m \times n}$, une matrice avec m lignes et n colonnes.
  - Matrices spéciales
    - **Matrice identité** : $\mathbf{I}$, où $\mathbf{I}_{ii} = 1$ et $\mathbf{I}_{ij} = 0$ pour $i \neq j$.
    - **Matrice diagonale** : $\text{diag}(\mathbf{v})$, où seuls les éléments diagonaux sont non nuls.

Variables Aléatoires et Opérations de base sur les Matrices

Variables Aléatoires et Opérations de base sur les Matrices

- **Les variables aléatoires scalaires** (représentées par des lettres majuscules en italique)
  - Notation : X, Y, Z.
  - Dimension : $X \in \mathbb{R}$ est une variable aléatoire scalaire réelle.
  - Exemple : $Y = f(X) + \mathcal{E}$, une variable aléatoire Y dépend d'une transformation f de la variable aléatoire X et d'un terme d'erreur $\mathcal{E}$.
- **Opérations de base sur les matrices et symboles clés**
  - **Transpose** : $\mathbf{A}^{\mathrm{T}}$, la transposée d'une matrice $\mathbf{A}$.
  - **Inverse** : $\mathbf{A}^{-1}$, l'inverse d'une matrice carrée $\mathbf{A}$, si elle existe.
  - **Produit scalaire** : $\mathbf{u}^{\mathrm{T}}\mathbf{v} = \sum_i u_i v_i$, le produit scalaire des vecteurs $\mathbf{u}$ et $\mathbf{v}$.
  - **Norme** : $||\mathbf{v}||$, la longueur ou la magnitude d'un vecteur.
  - **Trace** : $\mathrm{Tr}(\mathbf{A}) = \sum_i \mathbf{A}_{ii}$, la somme des éléments diagonaux d'une matrice carrée $\mathbf{A}$.
  - **Déterminant** : $|\mathbf{A}|$ ou $\det(\mathbf{A})$, un scalaire qui indique le facteur d'échelle de la transformation décrite par $\mathbf{A}$.

Notation Spécifique à la Régression

Notation Spécifique à la Régression

- **Notation spécifique à la régression**
  - $\mathbf{X}$: Matrice de conception de dimension $n \times (p+1)$, où :
    - n est le nombre d'observations
    - p est le nombre de prédicteurs.
  - $\mathbf{y}$: Vecteur des réponses de dimension n.
  - $\boldsymbol{\beta}$: Vecteur des coefficients de dimension $(p+1)$.
  - $\boldsymbol{\mathcal{E}}$: Vecteur des erreurs de dimension n.
  - $\hat{\boldsymbol{\beta}}$: L'estimateur du vecteur des coefficients $\boldsymbol{\beta}$.
    - Exemple : Estimateur des moindres carrés ordinaires (OLS, de l'anglais Ordinary Least squares)

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Notation Spécifique à la Régression

Ce qu'il Faut Retenir

- **Ce qu'il faut retenir**
  - **Constantes** : Lettres minuscules en italique (par ex. a, b).
  - **Vecteurs** : Lettres minuscules en gras (par ex. $\mathbf{x}, \mathbf{y}$).
  - **Matrices** : Lettres majuscules en gras (par ex. $\mathbf{X}, \mathbf{A}$).
  - **Variables aléatoires scalaires** : Lettres majuscules en italique (par ex. X, Y).

Table des Matières

Introduction : Pourquoi l'Algèbre Linéaire ?

Introduction : Pourquoi l'Algèbre Linéaire ?

- **Base mathématique :** L'algèbre linéaire constitue la base des concepts mathématiques utilisés pour :
  - Exprimer des modèles de régression.
  - Réduire la dimension de la matrice de conception $\mathbf{X}$:
    - Analyse en compsantes principales (ACP).
    - Décomposition en valeurs singulières (SVD, de l'anglais Singular Value Decomposition)
  - En réseaux de neurones et pour l'optimisation des modèles.
  - etc.

Introduction : Pourquoi l'Algèbre Linéaire ?

- **Application: Représentations matricielles pour la modèlisation**
  - Considérons le modèle de régression linéaire simple:

$$Y = \beta_0 + \beta_1 X + \mathcal{E}$$

  - En notation matricielle :

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\mathcal{E}}$$

où :

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}, \quad \boldsymbol{\mathcal{E}} = \begin{bmatrix} \mathcal{E}_0 \\ \mathcal{E}_1 \\ \vdots \\ \mathcal{E}_n \end{bmatrix}$$

Introduction : Pourquoi l'Algèbre Linéaire ?

- **Application: Résolution des systèmes d'équations**
  - Pour estimer $\boldsymbol{\beta}$, nous résolvons :

$$\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^T\mathbf{y}$$

  - Exemple : Avec

$$\mathbf{X} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 2 \\ 2.5 \\ 3.5 \end{bmatrix}$$

    on calcule l'estimateur de $\boldsymbol{\beta}$:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- $\Rightarrow$ L'algèbre linéaire permet **la modélisation et calcul efficace pour les données de haute dimension**.

Vecteurs et Espaces Vectoriels

Vecteurs

Vecteurs

- Un **vecteur** est une collection de nombres représentant :
  - Des points dans l'espace (par exemple, coordonnées en 2D ou 3D).
  - Des quantités avec une magnitude et une direction.
  - Des points de données (observations).
- **Représentation mathématique**

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}, \quad \text{où } v_i \in \mathbb{R} \Rightarrow \mathbf{v} \in \mathbb{R}^n.$$

Espace Vectoriels

Espace Vectoriel $\mathbb{R}^n$

Soient $\mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}$, $\mathbf{v} = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}$, et $\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}$ des vecteurs de $\mathbb{R}^n$, et $\lambda, \mu \in \mathbb{R}$.

L'espace vectoriel $\mathbb{R}^n$ est un ensemble de vecteurs où l'addition vectorielle et la multiplication scalaire respectent les propriétés suivantes :

- **Commutativité de la somme** : $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$.
- **Associativité de la somme** : $(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$.
- **0 élément neutre pour la somme** : $\mathbf{u} + 0 = 0 + \mathbf{u} = \mathbf{u}$.
- **Existence d'un inverse pour la somme** : $\mathbf{u} + (-\mathbf{u}) = \mathbf{0}$.

Espace Vectoriel $\mathbb{R}^n$

- 1 **est l'élément neutre pour le produit (par un scalaire)** : $1 \cdot \mathbf{u} = \mathbf{u}$.
- **Associativité du produit (par un scalaire)** : $\lambda \cdot (\mu \cdot \mathbf{u}) = (\lambda\mu) \cdot \mathbf{u}$.
- **Distributivité du produit par rapport à la somme**: $\lambda \cdot (\mathbf{u} + \mathbf{v}) = \lambda \cdot \mathbf{u} + \lambda \cdot \mathbf{v}$.
- **Distributivité de la somme par rapport au produit** : $(\lambda + \mu) \cdot \mathbf{u} = \lambda \cdot \mathbf{u} + \mu \cdot \mathbf{u}$.

Ces 8 propriétés définissent ce qu'est un espace vectoriel
$\Rightarrow \mathbb{R}^n$ est un **espace vectoriel**.

Sous-Espace Vectoriel de $\mathbb{R}^n$

F est appelée un **sous-espace vectoriel** $\mathbb{R}^n$ si F satisfait les propriétés suivantes :

- Le vecteur nul $\mathbf{0}$ appartient à F.
- La somme de deux vecteurs $\mathbf{u}, \mathbf{v} \in F$ appartient également à F $(\mathbf{u} + \mathbf{v} \in F)$.
- Le produit d'un vecteur $u \in F$ par un scalaire $\lambda \in \mathbb{R}$ appartient à F $(\lambda \cdot \mathbf{u} \in F)$.

Ces propriétés garantissent que F est fermé pour les opérations de somme vectorielle et de multiplication scalaire, et donc que F est un sous-espace vectoriel de $\mathbb{R}^n$.

Opérations sur les Vecteurs

Opérations de Base sur les Vecteurs

Opérations de Base sur les Vecteurs

- **Addition**

$$\mathbf{u} + \mathbf{v} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} u_1 + v_1 \\ u_2 + v_2 \end{bmatrix}.$$

- **Multiplication Scalaire**

$$\alpha\mathbf{v} = \alpha \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \alpha v_1 \\ \alpha v_2 \end{bmatrix}.$$

- **Produit Scalaire**

$$\mathbf{u}^{\mathrm{T}}\mathbf{v} = \sum_{i=1}^{n} u_i v_i.$$

Opérations de Base sur les Vecteurs: Exemples Numériques

Considérons les vecteurs suivants :

$$\mathbf{u} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, \quad \alpha = 2.$$

- **Addition**

$$\mathbf{u} + \mathbf{v} = \begin{bmatrix} 3 \\ -2 \end{bmatrix} + \begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 3+1 \\ -2+4 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}.$$

- **Multiplication Scalaire**

$$\alpha\mathbf{v} = 2 \begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 2 \cdot 1 \\ 2 \cdot 4 \end{bmatrix} = \begin{bmatrix} 2 \\ 8 \end{bmatrix}.$$

- **Produit Scalaire**

$$\mathbf{u} \cdot \mathbf{v} = \begin{bmatrix} 3 \\ -2 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 4 \end{bmatrix} = (3 \cdot 1) + (-2 \cdot 4) = 3 - 8 = -5.$$

Opérations de Base sur les Vecteurs : Code Python

```python
import numpy as np

# Definir les vecteurs
u = np.array([1, 2])
v = np.array([3, 4])

# Addition
print("Addition de vecteurs :", u + v)

# Multiplication scalaire
alpha = 2
print("Multiplication scalaire :", alpha * u)

# Produit scalaire
print("Produit scalaire :", np.dot(u, v))
```

Résultats:

```
Addition de vecteurs : [4 6]
Multiplication scalaire : [2 4]
Produit scalaire : 11
```

Produit Extérieur (Outer Product)

Produit Extérieur (Outer Product)

- Produit Extérieur (Outer Product)
  - Soient $\mathbf{x} \in \mathbb{R}^m$ et $\mathbf{y} \in \mathbb{R}^n$ (pas nécessairement de la même taille), leur **produit extérieur** $\mathbf{x}\mathbf{y}^{\mathrm{T}}$ est une matrice de dimensions $m \times n$ dont les entrées sont données par :

$$(\mathbf{x}\mathbf{y}^{\mathrm{T}})_{ij} = x_i y_j.$$

Matriciellement, cela s'écrit :

$$\mathbf{x}\mathbf{y}^{\mathrm{T}} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix} = \begin{bmatrix} x_1 y_1 & x_1 y_2 & \cdots & x_1 y_n \\ x_2 y_1 & x_2 y_2 & \cdots & x_2 y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_m y_1 & x_m y_2 & \cdots & x_m y_n \end{bmatrix}.$$

Produit Extérieur (Outer Product)

- **Exemple d'utilisation**
  Considérons $\mathbf{1} \in \mathbb{R}^n$ comme un vecteur dont toutes les composantes sont égales à 1. Si nous prenons une matrice $\mathbf{A} \in \mathbb{R}^{m \times n}$ où chaque colonne est égale à un vecteur $\mathbf{x} \in \mathbb{R}^m$, le produit extérieur permet de représenter cette matrice :

$$\mathbf{A} = \mathbf{x}\mathbf{1}^{\mathrm{T}}.$$

Matriciellement, cela s'écrit :

$$\mathbf{x}\mathbf{1}^{\mathrm{T}} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}}_{n \text{ colonnes}} = \underbrace{\begin{bmatrix} x_1 & x_1 & \cdots & x_1 \\ x_2 & x_2 & \cdots & x_2 \\ \vdots & \vdots & \ddots & \vdots \\ x_m & x_m & \cdots & x_m \end{bmatrix}}_{n \text{ colonnes}}.$$

Produit Extérieur (Outer Product) : Code Python

```python
import numpy as np

# Définir les vecteurs
x = np.array([2, 3, 4])  # Vecteur x
ones = np.ones(5)        # Vecteur de 1 avec 5 éléments

# Calcul du produit extérieur
outer_product = np.outer(x, ones)

# Affichage des résultats
print("Vecteur x :", x)
print("Vecteur de 1 :", ones)
print("Produit extérieur :")
print(outer_product)
```

Résultats:

```
Vecteur x : [2 3 4]
Vecteur de 1 : [1. 1. 1. 1. 1.]
Produit extérieur :
[[2. 2. 2. 2. 2.]
 [3. 3. 3. 3. 3.]
 [4. 4. 4. 4. 4.]]
```

Norme d'un Vecteur

Norme d'un Vecteur

- **Définition** : La norme d'un vecteur $\mathbf{v} \in \mathbb{R}^n$, notée par $\|\mathbf{v}\|$, est une mesure de sa longueur dans l'espace vectoriel. Par exemple, la norme Euclidienne ℓ_2 d'un vecteur $\mathbf{v} \in \mathbb{R}^n$ est définie par :

$$\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2}.$$

- **Propriétés** :
  - $\|\mathbf{v}\| \geq 0$, avec $\|\mathbf{v}\| = 0$ si $\mathbf{v} = \mathbf{0}$.
  - $\|\alpha\mathbf{v}\| = |\alpha|\|\mathbf{v}\|$ pour $\alpha \in \mathbb{R}$.
  - $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$ (inégalité triangulaire).

Normes ℓ_p

- Les normes ℓ_p sont définies par :

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}, \quad p \geq 1.$$

- **Exemples de normes ℓ_p** (Exemple Numérique, $\mathbf{v} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$) :
  - **Norme ℓ_1 (Manhattan)** :

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|. \quad \Rightarrow \|\mathbf{v}\|_1 = |3| + |4| = 7.$$

  - **Norme ℓ_2 (Euclidienne)** :

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}. \quad \Rightarrow \|\mathbf{v}\|_2 = \sqrt{3^2 + 4^2} = 5.$$

Norme ℓ_∞ (Infinie) et Code Python pour les Normes

- **La norme** ℓ_∞ est définie par :

$$\|\mathbf{x}\|_\infty = \max_i |x_i|. \quad \Rightarrow \|\mathbf{v}\|_\infty = \max(|3|, |4|) = 4.$$

```python
import numpy as np
# Définir le vecteur
v = np.array([3, 4])
# Calculer les normes
euclidean_norm = np.linalg.norm(v, ord=2)
manhattan_norm = np.linalg.norm(v, ord=1)
infinity_norm = np.linalg.norm(v, ord=np.inf)
# Afficher les résultats
print("Vecteur:", v)
print("Norme Euclidienne:", euclidean_norm)
print("Norme Manhattan:", manhattan_norm)
print("Norme infinie:", infinity_norm)
```

Résultats:

```
Vecteur: [3 4]
Norme Euclidienne: 5.0
Norme Manhattan: 7.0
Norme infinie: 4.0
```

Visualisation des Normes ℓ_1, ℓ_2 et ℓ_∞

- Les normes ℓ_1, ℓ_2, et ℓ_∞ peuvent être visualisées sous forme de sphères unités (ensemble des points $\mathbf{x} \in \mathbb{R}^2$ tels que $\|\mathbf{x}\| = 1$) :
  - La norme ℓ_1 privilégie les déplacements alignés avec les axes, ce qui se traduit par un losange.
  - La norme ℓ_2, la plus courante, mesure les distances euclidiennes. La sphère d'unité est lisse et circulaire, représentant un poids égal dans toutes les directions (cercle).
  - La norme ℓ_∞ prend la valeur maximale des composantes, donnant lieu à un carré.

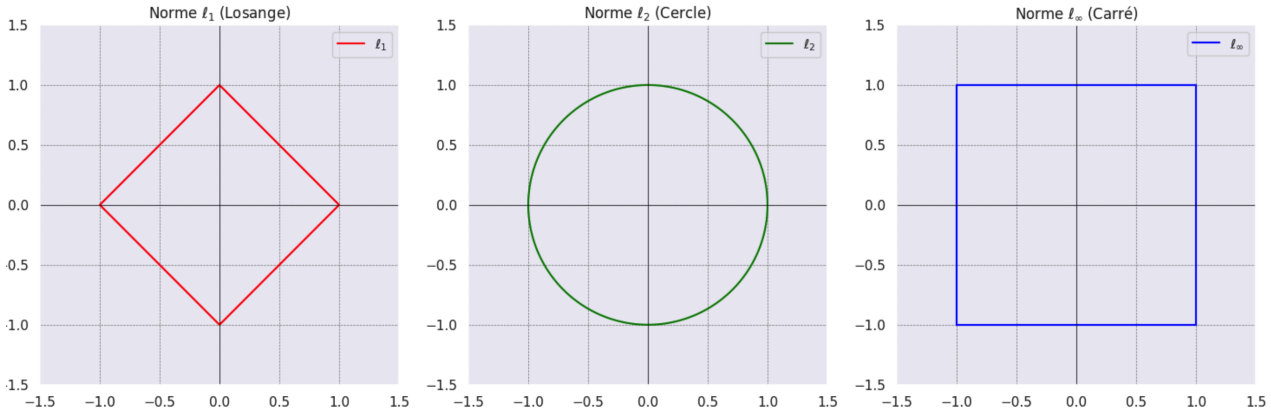


Figure: Représentation des normes ℓ_1 (losange), ℓ_2 (cercle), et ℓ_∞ (carré).

Intuition des Normes : Norme ℓ_1 (Norme Manhattan)

- **Norme ℓ_1 (Norme Manhattan)** :
  - **Interprétation** : Mesure la distance comme si l'on se déplaçait à travers un réseau orthogonal, comme les rues d'une ville (Manhattan).
  - **Application** : Utilisée dans le modèle de régression Lasso pour exprimer la régularisation L_1. Elle permet d'obtenir des paramètres éparses.
  - **Propriété** : Robuste aux données éparses (sparse data), elle encourage les solutions où de nombreux paramètres sont nuls. Elle effectue une sélection des caractéristiques en éliminant celles qui contribuent le moins au modèle.

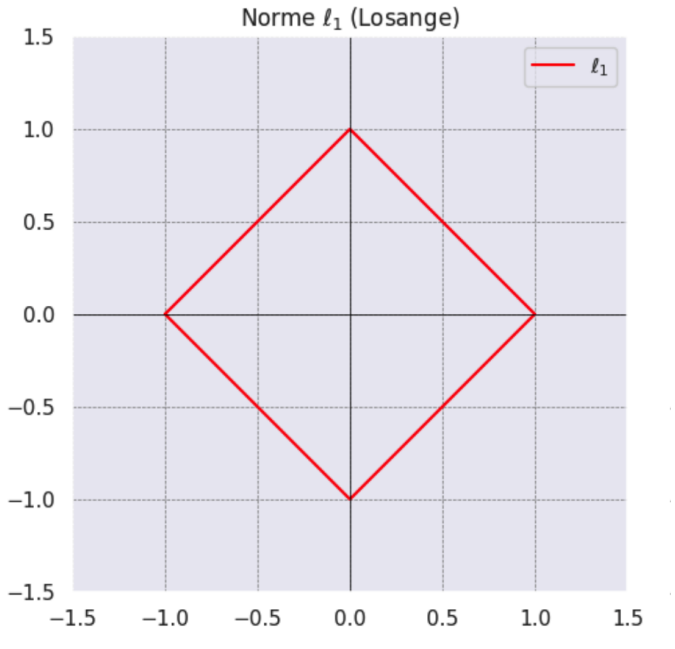


Figure: Norme ℓ_1 : Sphère unité (forme de losange)

Intuition des Normes : Norme ℓ_2 (Norme Euclidienne)

- **Norme ℓ_2 (Norme Euclidienne)** :
  - **Interprétation** : mesure la distance droite (ou longueur) entre deux points dans un espace euclidien. C'est la norme classique que l'on utilise intuitivement pour mesurer une distance.
  - **Application** : Utilisée dans la régression Ridge utilise pour exprimer régularisation L_2 pour pénaliser les valeurs extrêmes des estimateurs.
  - **Propriété** : Utilisée pour la régularisation L_2, elle ne réalise pas de sélection de caractéristiques, mais équilibre les contributions de toutes les caractéristiques en les réduisant proportionnellement.

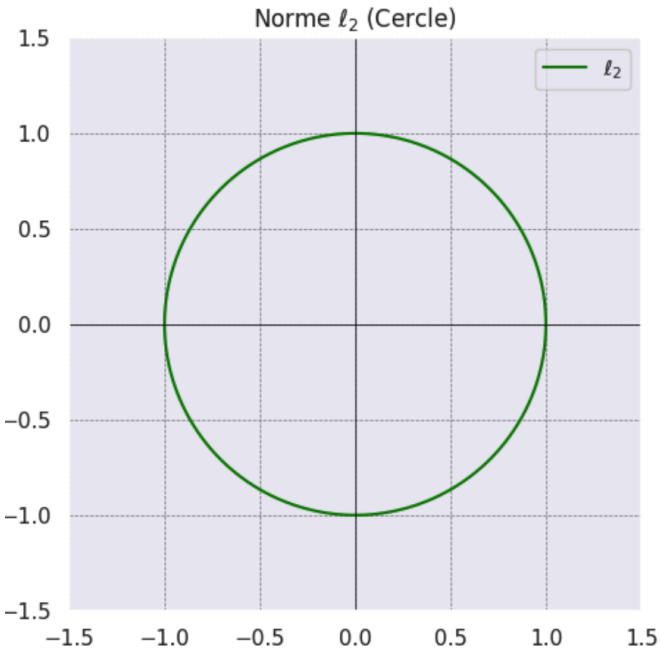


Figure: Norme ℓ_2 : Sphère unité (forme de cercle)

Intuition des Normes : Norme ℓ_∞

- **Norme ℓ_∞ (Norme infinie)** :
  - **Interprétation** : La norme ℓ_∞ mesure la plus grande valeur absolue parmi les éléments d'un vecteur. Elle reflète l'impact maximal d'un seul élément.
  - **Application** : Utile lorsqu'on veut minimiser ou contrôler le plus grand écart ou erreur dans un ensemble de données.
    - **Exemple** : Utilisée en optimisation des pires cas (worst-case optimization).
  - **Note**: Utile dans les contextes où des valeurs extrêmes peuvent entraîner des problèmes.

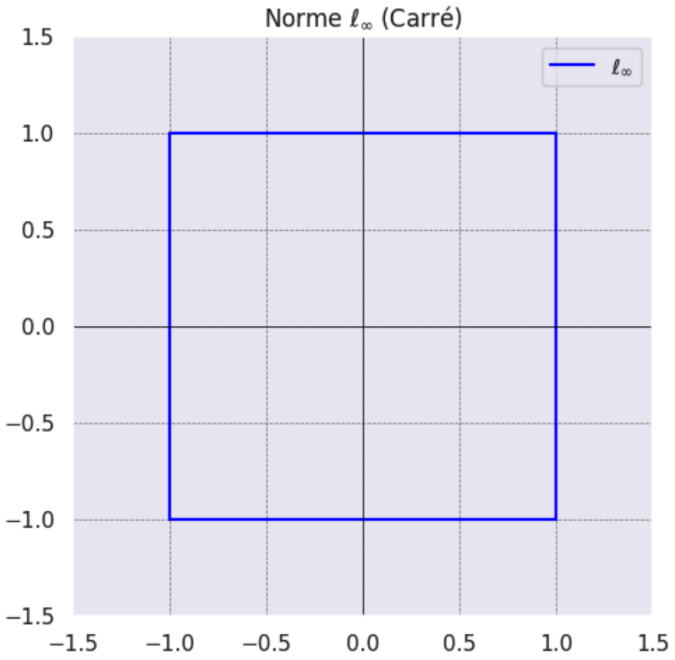


Figure: Norme ℓ_∞ : Sphère unité (forme de carré)

Opérations sur les Matrices

Addition de Matrices

Addition de Matrices

- **Addition de Matrices** : Soient $\mathbf{A}$ et $\mathbf{B}$ deux matrices de même dimension $\mathbb{R}^{m \times n}$. $\mathbf{A}$ et $\mathbf{B}$ peuvent être alors additionnées tel que:

$$\mathbf{C} = \mathbf{A} + \mathbf{B}, \quad \text{où } c_{ij} = a_{ij} + b_{ij}.$$

- **Propriétés** :
  - **Commutativité :** $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$.
  - **Associativité :** $(\mathbf{A} + \mathbf{B}) + \mathbf{C} = \mathbf{A} + (\mathbf{B} + \mathbf{C})$.
  - **Élément neutre** $\mathbf{0}$, (la matrice nulle): $\mathbf{A} + \mathbf{0} = \mathbf{A}$.
  - **Inverse :** Chaque matrice $\mathbf{A}$ a une matrice opposée $-\mathbf{A}$ telle que :

$$\mathbf{A} + (-\mathbf{A}) = \mathbf{0}.$$

  - **Clôture :** Si $\mathbf{A}$ et $\mathbf{B}$ ont la même dimension $m \times n$, alors $\mathbf{A} + \mathbf{B}$ est aussi de dimension $m \times n$.

Addition de Matrices : Exemple Numérique

Considérons les matrices suivantes :

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}.$$

- **Addition** :

$$\mathbf{C} = \mathbf{A} + \mathbf{B} = \begin{bmatrix} 1+7 & 2+8 & 3+9 \\ 4+10 & 5+11 & 6+12 \end{bmatrix} = \begin{bmatrix} 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix}.$$

Addition de Matrices : Code Python

```python
import numpy as np

# Définir les matrices
A = np.array([[1, 2, 3], [4, 5, 6]])
B = np.array([[7, 8, 9], [10, 11, 12]])

# Addition de matrices
C = A + B
print("Addition de matrices:\n", C)
```

Résultat :

```
Addition de matrices:
 [[ 8 10 12]
 [14 16 18]]
```

Multiplication Scalaire des Matrices

Multiplication Scalaire des Matrices

- **Multiplication Scalaire** : Un scalaire α multiplie chaque élément de la matrice $\mathbf{A}$:

$$\mathbf{B} = \alpha\mathbf{A}, \quad \text{où } b_{ij} = \alpha a_{ij}.$$

- **Propriétés** :
  - **Distributivité :** $\alpha(\mathbf{A} + \mathbf{B}) = \alpha\mathbf{A} + \alpha\mathbf{B}$ et $(\alpha + \beta)\mathbf{A} = \alpha\mathbf{A} + \beta\mathbf{A}$.
  - **Associativité :** $\alpha(\beta\mathbf{A}) = (\alpha\beta)\mathbf{A}$.
  - **Élément neutre :** $1 \cdot \mathbf{A} = \mathbf{A}$.
  - **Propriété du zéro :** $0 \cdot \mathbf{A} = \mathbf{0}$, où $\mathbf{0}$ est une matrice nulle de même dimension que $\mathbf{A}$.
  - **Clôture :** Si $\mathbf{A}$ est une matrice et α est un scalaire, alors $\alpha\mathbf{A}$ est une matrice de mêmes dimensions que $\mathbf{A}$.

Multiplication Scalaire des Matrices : Exemple Numérique

Considérons la matrice suivante :

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}.$$

- **Multiplication Scalaire** :

$$\mathbf{B} = \alpha \mathbf{A}, \quad \text{où } b_{ij} = \alpha a_{ij}.$$

Par exemple, si $\alpha = 3$, alors :

$$\mathbf{B} = 3 \cdot \mathbf{A} = \begin{bmatrix} 3 \cdot 1 & 3 \cdot 2 & 3 \cdot 3 \\ 3 \cdot 4 & 3 \cdot 5 & 3 \cdot 6 \end{bmatrix} = \begin{bmatrix} 3 & 6 & 9 \\ 12 & 15 & 18 \end{bmatrix}.$$

Multiplication Scalaire des Matrices : Code Python

```python
import numpy as np

# Définir la matrice
A = np.array([[1, 2, 3], [4, 5, 6]])

# Scalaire
alpha = 3

# Multiplication scalaire
B = alpha * A
print("Multiplication scalaire de la matrice:\n", B)
```

Résultat :

```
Multiplication scalaire de la matrice:
 [[ 3  6  9]
 [12 15 18]]
```

Produit Matrice-Vecteur

Produit Matrice-Vecteur

- **Produit Matrice-Vecteur**: Le produit d'une matrice $\mathbf{A} \in \mathbb{R}^{m \times n}$ avec un vecteur $\mathbf{x} \in \mathbb{R}^n$ donne un vecteur $\mathbf{y} \in \mathbb{R}^m$, tel que :

$$\mathbf{y} = \mathbf{A}\mathbf{x}.$$

- Chaque élément y_i du vecteur résultat $\mathbf{y}$ est égal au produit scalaire de la i-ème ligne de $\mathbf{A}$ avec $\mathbf{x}$:

$$y_i = \mathbf{a}_i^T \mathbf{x}.$$

- **Représentation par colonnes :** En écrivant $\mathbf{A}$ en termes de ses colonnes $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$, nous avons :

$$\mathbf{y} = \mathbf{A}\mathbf{x} = x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \cdots + x_n\mathbf{a}_n.$$

Ici, le produit est une *combinaison linéaire* des colonnes de $\mathbf{A}$, pondérée par les éléments de $\mathbf{x}$.

Produit Matrice-Vecteur : Exemple Numérique

- Soient la matrice $\mathbf{A}$ et le vecteur $\mathbf{x}$ suivants :

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}.$$

- Le produit $\mathbf{y} = \mathbf{A}\mathbf{x}$ est donné par :

$$\mathbf{y} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \cdot 1 + 2 \cdot 0 + 3 \cdot (-1) \\ 4 \cdot 1 + 5 \cdot 0 + 6 \cdot (-1) \\ 7 \cdot 1 + 8 \cdot 0 + 9 \cdot (-1) \end{bmatrix} = \begin{bmatrix} -2 \\ -2 \\ -2 \end{bmatrix}.$$

Produit Matrice-Vecteur: Code Python

```python
import numpy as np

# Définir la matrice A et le vecteur x
A = np.arange(1,10,1).reshape((3,3))
x = np.array([1, 0, -1])

# Calculer le produit matrice-vecteur
y = np.dot(A, x)

# Afficher les résultats
print("Matrice A :\n", A)
print("Vecteur x :", x)
print("Produit Matrice-Vecteur (y = A x) :", y)
```

Résultats:

```
Matrice A :
 [[1 2 3]
  [4 5 6]
  [7 8 9]]
Vecteur x : [ 1  0 -1]
Produit Matrice-Vecteur (y = A x) : [-2 -2 -2]
```

Produit Matrice-Matrice

Produit Matrice-Matrice

- **Produit Matrice-Matrice**: Soient deux matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ et $\mathbf{B} \in \mathbb{R}^{n \times p}$, leur produit est une matrice $\mathbf{C} \in \mathbb{R}^{m \times p}$ telle que chaque élément c_{ij} de $\mathbf{C}$ est donné par :

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.$$

- $\mathbf{C} = \mathbf{AB}$ peut aussi être vu comme une combinaison linéaire des colonnes de $\mathbf{B}$ pondérées par les lignes de $\mathbf{A}$.
- Les matrices sont définies comme suit :

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix}.$$

Produit Matrice-Matrice : Exemple Numérique

- Le produit $\mathbf{C} = \mathbf{AB}$ est donné par :

$$\mathbf{C} = \begin{bmatrix} \sum_{k=1}^{n} a_{1k}b_{k1} & \sum_{k=1}^{n} a_{1k}b_{k2} & \cdots & \sum_{k=1}^{n} a_{1k}b_{kp} \\ \sum_{k=1}^{n} a_{2k}b_{k1} & \sum_{k=1}^{n} a_{2k}b_{k2} & \cdots & \sum_{k=1}^{n} a_{2k}b_{kp} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{k=1}^{n} a_{mk}b_{k1} & \sum_{k=1}^{n} a_{mk}b_{k2} & \cdots & \sum_{k=1}^{n} a_{mk}b_{kp} \end{bmatrix}.$$

- Considérons les matrices suivantes :

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 7 & 8 \\ 9 & 10 \end{bmatrix}.$$

- Le produit $\mathbf{C} = \mathbf{AB}$ est donné par :

$$\mathbf{C} = \begin{bmatrix} 1 \cdot 7 + 2 \cdot 9 & 1 \cdot 8 + 2 \cdot 10 \\ 3 \cdot 7 + 4 \cdot 9 & 3 \cdot 8 + 4 \cdot 10 \\ 5 \cdot 7 + 6 \cdot 9 & 5 \cdot 8 + 6 \cdot 10 \end{bmatrix} = \begin{bmatrix} 18 & 28 \\ 57 & 64 \\ 89 & 100 \end{bmatrix}.$$

Produit Matrice-Matrice : Code Python

```python
import numpy as np
# Définir les matrices A et B
A = np.arange(1,7).reshape((3,2))
B = np.arange(7,11).reshape((2,2))
# Calculer le produit matrice-matrice
C = np.dot(A, B)
# Afficher les résultats
print("Matrice A :\n", A)
print("Matrice B :\n", B)
print("Produit Matrice-Matrice (C = A x B) :\n", C)
```

Résultats:

```
Matrice A :
 [[1 2]
  [3 4]
  [5 6]]
Matrice B :
 [[7 8]
  [9 10]]
Produit Matrice-Matrice (C = A x B) :
 [[18 28]
  [57 64]
  [89 100]]
```

La Matrice Identité et les Matrices Diagonales

La Matrice Identité

- **Définition** : La matrice identité $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ est une matrice carrée telle que :

$$(\mathbf{I}_n)_{ij} = \begin{cases} 1 & \text{si } i = j, \\ 0 & \text{si } i \neq j. \end{cases} \quad , \quad \mathbf{I} = \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}}_{n \text{ colonnes}}.$$

- **Propriété fondamentale** :

$$\mathbf{I}_n \mathbf{A} = \mathbf{A}, \quad \mathbf{A}\mathbf{I}_n = \mathbf{A}, \quad \text{pour toute matrice } \mathbf{A}.$$

Les Matrices Diagonales

- **Définition** : Une matrice diagonale $\mathbf{D} \in \mathbb{R}^{n \times n}$ est une matrice carrée où tous les éléments hors de la diagonale principale sont nuls :

$$(\mathbf{D})_{ij} = \begin{cases} d_i & \text{si } i = j, \\ 0 & \text{si } i \neq j. \end{cases}$$

$$\mathbf{D} = \text{diag}(\mathbf{d}) = \underbrace{\begin{bmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{bmatrix}}_{n \text{ colonnes}}, \quad \mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix}.$$

La Matrice Identité et Matrices Diagonales : Code Python

```python
import numpy as np

# Matrice Identité
I = np.eye(3)
print("Matrice Identité :\n", I)

# Matrice Diagonale
d = [3, 5, 7]
D = np.diag(d)
print("Matrice Diagonale :\n", D)
```

Résultats:

```
Matrice Identité :
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
Matrice Diagonale :
[[3 0 0]
 [0 5 0]
 [0 0 7]]
```

Transposée d'une Matrice

Transposée d'une Matrice

- **Définition** : La transposée d'une matrice $\mathbf{A} \in \mathbb{R}^{m \times n}$ est notée $\mathbf{A}^{\mathrm{T}}$. Elle est obtenue en échangeant les lignes et les colonnes de $\mathbf{A}$ tel que $\mathbf{A}^{\mathrm{T}} \in \mathbb{R}^{n \times m}$:

$$(\mathbf{A}^{\mathrm{T}})_{ij} = \mathbf{A}_{ji}.$$

- **Exemple** :

$$\mathbf{A} = \underbrace{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}}_{m \text{ colonnes}}, \quad \mathbf{A}^{\mathrm{T}} = \underbrace{\begin{bmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1m} & a_{2m} & \cdots & a_{nm} \end{bmatrix}}_{n \text{ colonnes}}.$$

Transposée d'une Matrice : Exemple Numérique

- **Propriétés** :
- $(\mathbf{A}^{\mathrm{T}})^{\mathrm{T}} = \mathbf{A}$.
- $(\alpha\mathbf{A})^{\mathrm{T}} = \alpha\mathbf{A}^{\mathrm{T}}$
- $(\mathbf{A}+\mathbf{B})^{\mathrm{T}} = \mathbf{A}^{\mathrm{T}} + \mathbf{B}^{\mathrm{T}}$ et $(\mathbf{A}-\mathbf{B})^{\mathrm{T}} = \mathbf{A}^{\mathrm{T}} - \mathbf{B}^{\mathrm{T}}$.
- $(\mathbf{AB})^{\mathrm{T}} = \mathbf{B}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}$.
- Considérons la matrice suivante :

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}.$$

La transposée est donnée par :

$$\mathbf{A}^{\mathrm{T}} = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}.$$

Transposée d'une Matrice : Code Python

```python
import numpy as np

# Définir la matrice
A = np.array([[1, 2, 3], [4, 5, 6]])

# Calculer la transposée
A_transpose = A.T

print("Matrice originale :\n", A)
print("Transposée de la matrice :\n", A_transpose)
```

Résultats:

```
Matrice originale :
[[1 2 3]
 [4 5 6]]
Transposée de la matrice :
[[1 4]
 [2 5]
 [3 6]]
```

Matrices Symétriques

Matrices Symétriques

- **Définition** : Une matrice $\mathbf{A} \in \mathbb{R}^{n \times n}$ est symétrique si elle est égale à sa transposée :

$$\mathbf{A}^{\mathrm{T}} = \mathbf{A}.$$

- **Propriétés** :
  - Les éléments d'une matrice symétrique sont symétriques par rapport à la diagonale principale :

$$a_{ij} = a_{ji}, \quad \forall i, j.$$

  - Pour toute matrice $\mathbf{B} \in \mathbb{R}^{n \times n}$, $\mathbf{A} = \mathbf{B} + \mathbf{B}^{\mathrm{T}}$ est symétrique.
  - La somme de deux matrices symétriques est symétrique.
- **Exemple** :

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}.$$

Cette matrice est symétrique car $a_{ij} = a_{ji}$ pour tout i, j.

Matrices Symétriques : Code Python

```python
import numpy as np

# Définir une matrice symétrique
A = np.array([[1, 2, 3],
              [2, 4, 5],
              [3, 5, 6]])

# Vérifier si la matrice est symétrique
is_symmetric = np.allclose(A, A.T)
print("Matrice A:\n", A)
print("A est-elle symétrique?:", is_symmetric)
```

Résultats:

```
Matrice A:
 [[1 2 3]
  [2 4 5]
  [3 5 6]]
A est-elle symétrique?: True
```

Trace d'une Matrice

Trace d'une matrice

- **Définition** : La trace d'une matrice carrée $\mathbf{A} \in \mathbb{R}^{n \times n}$ est la somme des éléments de sa diagonale principale :

$$\text{tr}(\mathbf{A}) = \sum_{i=1}^{n} a_{ii}.$$

- **Exemple** :

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad \Rightarrow \text{tr}(\mathbf{A}) = \sum_{i=1}^{n} a_{ii} = a_{11} + a_{22} + \cdots + a_{nn}.$$

Trace d'une Matrice

- **Propriétés**:
  - Pour $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\text{tr}(\mathbf{A}) = \text{tr}(\mathbf{A}^{\text{T}})$.
  - Pour $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$, $\text{tr}(\mathbf{A} + \mathbf{B}) = \text{tr}(\mathbf{A}) + \text{tr}(\mathbf{B})$.
  - Pour $\mathbf{A} \in \mathbb{R}^{n \times n}$ et $\alpha \in \mathbb{R}$, $\text{tr}(\alpha\mathbf{A}) = \alpha\,\text{tr}(\mathbf{A})$.
  - Pour $\mathbf{A}, \mathbf{B}$ telles que $\mathbf{AB}$ est une matrice carrée, $\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA})$.
  - Pour $\mathbf{A}, \mathbf{B}, \mathbf{C}$ telles que $\mathbf{ABC}$ est une matrice carrée :

$$\text{tr}(\mathbf{ABC}) = \text{tr}(\mathbf{BCA}) = \text{tr}(\mathbf{CAB}),$$

    et ainsi de suite pour le produit de plusieurs matrices.
  - $\text{tr}(\mathbf{A}^{\text{T}}\mathbf{B}) = \text{tr}(\mathbf{AB}^{\text{T}}) = \text{tr}(\mathbf{B}^{\text{T}}\mathbf{A}) = \text{tr}(\mathbf{BA}^{\text{T}}) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}b_{ij}$.
    $\Rightarrow$ Produit matriciel de Hadamard (élément par élément)

Trace d'une Matrice : Exemple Numérique

- Considérons la matrice suivante :

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

- La trace de $\mathbf{A}$ est la somme des éléments de la diagonale principale :

$$\text{tr}(\mathbf{A}) = 1 + 5 + 9 = 15.$$

Trace d'une Matrice : Code Python

```python
import numpy as np

# Définir la matrice
A = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

# Calculer la trace
trace_A = np.trace(A)

# Afficher le résultat
print("La trace de la matrice A est :", trace_A)
```

Résultat :

```
La trace de la matrice A est : 15
```

Le Déterminant d'une Matrice

Le Déterminant d'une Matrice

- **Définition :** Le déterminant d'une matrice carrée $\mathbf{A}$, noté $|\mathbf{A}|$ ou $\det(\mathbf{A})$, est une fonction scalaire des éléments de $\mathbf{A}$. Il indique le facteur d'échelle de la transformation décrite par $\mathbf{A}$.
  - **Formule récursive :**

$$\det(\mathbf{A}) = |\mathbf{A}| = \sum_{i=1}^{n} (-1)^{i+j} a_{ij} \det(\mathbf{A}_{\setminus i, \setminus j})$$

  - **Note :**
    - j est un entier tel que $j \in \{1, \ldots, n\}$.
    - $\mathbf{A}_{\setminus i, \setminus j}$ est la matrice obtenue en supprimant la i-ième ligne et la j-ième colonne de $\mathbf{A}$.

Le Déterminant d'une Matrice

- **Formule récursive :**

$$\det(\mathbf{A}) = |\mathbf{A}| = \sum_{i=1}^{n} (-1)^{i+j} a_{ij} \det(\mathbf{A}_{\setminus i, \setminus j})$$

- **Note :**
  - j est un entier tel que $j \in \{1, \ldots, n\}$.
  - $\mathbf{A}_{\setminus i, \setminus j}$ est la matrice obtenue en supprimant la i-ième ligne et la j-ième colonne de $\mathbf{A}$.
- **Calcul du déterminant pour les matrices 1×1 et 2×2 :**

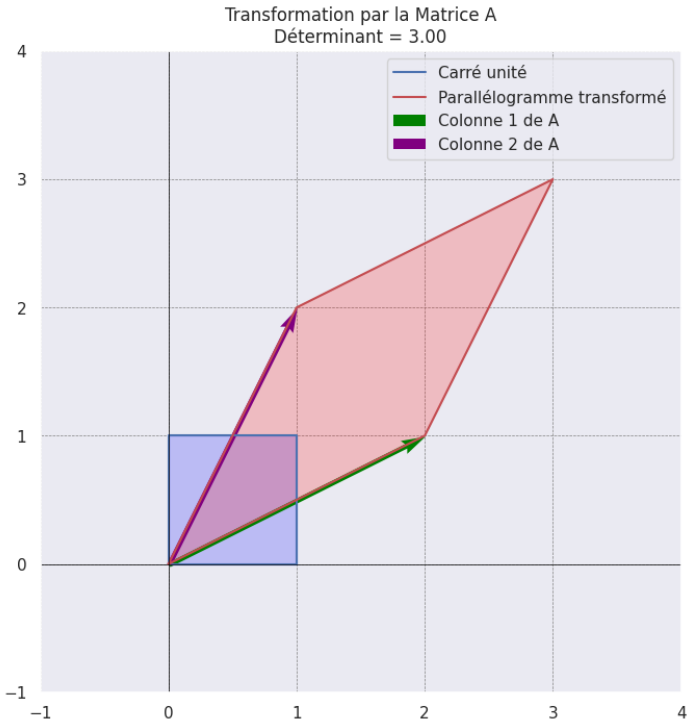
$$\left|a_{11}\right| = a_{11}, \quad \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}.$$

- **Formule pour une matrice 3×3 :**

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a(ei - hf) - b(di - gf) + c(dh - eg).$$

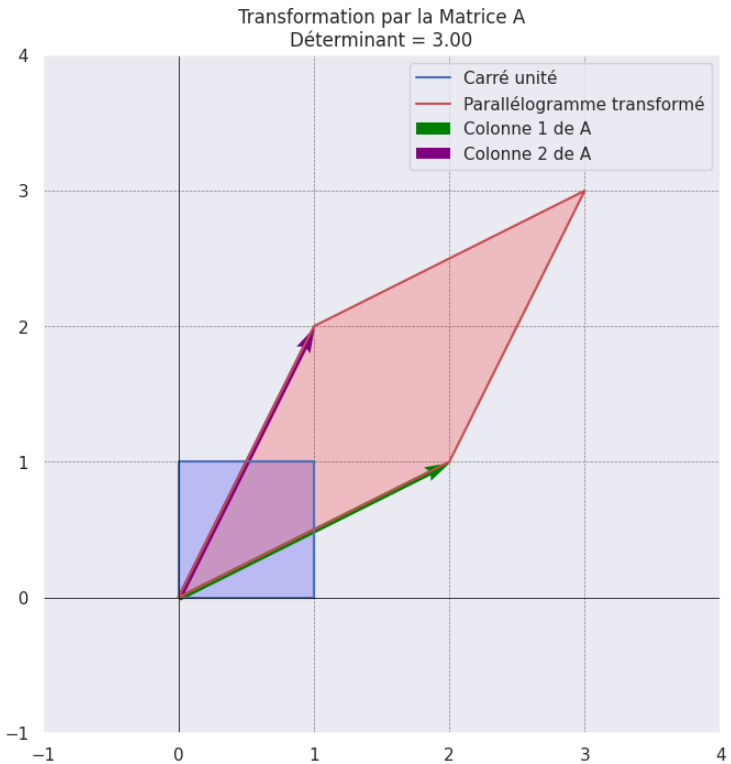
Le Déterminant d'une Matrice: Interprétation Géométrique

- Pour $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\det(\mathbf{A})$ décrit le facteur d'échelle de la transformation linéaire définie par $\mathbf{A}$.
- $\mathbf{A} \in \mathbb{R}^{2 \times 2}$, $\det(\mathbf{A})$ correspond à l'aire signée du parallélogramme formé par les colonnes de $\mathbf{A}$. Par exemple, $\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$



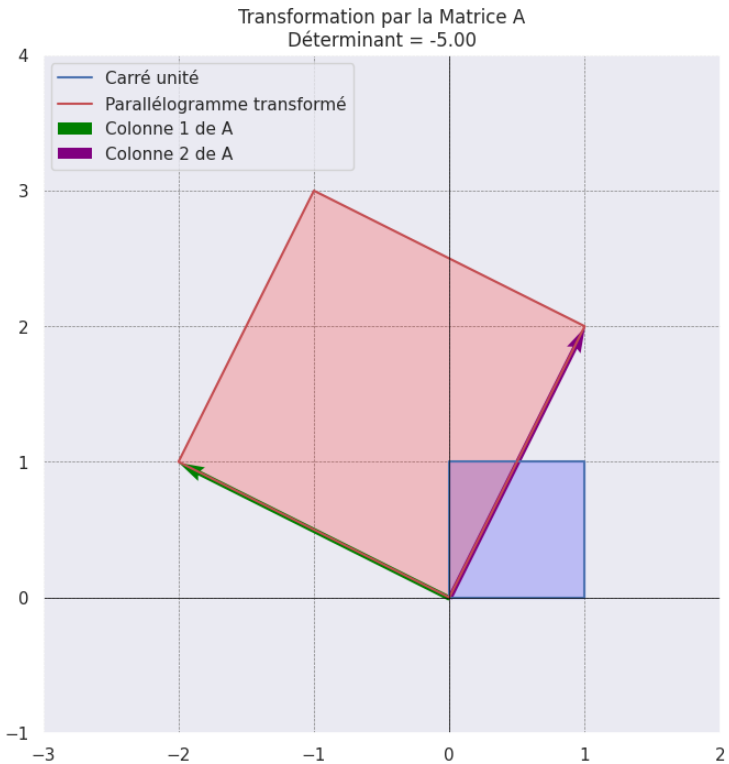
Le Déterminant d'une Matrice: Interprétation Géométrique

- Si $\det(\mathbf{A}) > 0$ l'orientation est conservée.
  - Par exemple, $\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \Rightarrow \det(\mathbf{A}) = 3$



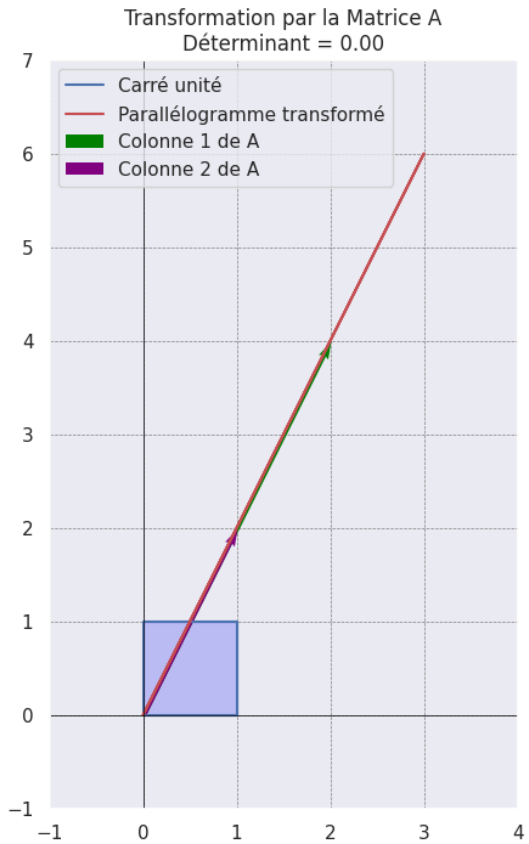
Le Déterminant d'une Matrice: Interprétation Géométrique

- Si $\det(\mathbf{A}) < 0$ l'orientation est inversée.
  - Par exemple, $\mathbf{A} = \begin{bmatrix} -2 & 1 \\ 1 & 2 \end{bmatrix} \Rightarrow \det(\mathbf{A}) = -5$



Le Déterminant d'une Matrice: Interprétation Géométrique

- Si $\det(\mathbf{A}) = 0$ cela signifie que la matrice "aplatie" la transformation en une dimension inférieure.
  - Par exemple, $\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 4 & 2 \end{bmatrix} \Rightarrow \det(\mathbf{A}) = 0$



Calcul du Déterminant avec Python

```python
import numpy as np

# Matrice 2 * 2
A = np.array([[4, 3],
              [6, 3]])

# Calcul du déterminant
det_A = np.linalg.det(A)
print(f"Déterminant de A : {det_A:.2f}")

# Matrice 3 * 3
B = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

# Calcul du déterminant
det_B = np.linalg.det(B)
print(f"Déterminant de B : {det_B:.2f}")
```

Résultat :

```
Déterminant de A : -6.00
Déterminant de B : 0.00
```

Norme de Frobenius pour les Matrices

Norme de Frobenius pour les Matrices

- **Définition** : La norme de Frobenius $\|\mathbf{A}\|_F$ pour une matrice $\mathbf{A} \in \mathbb{R}^{m \times n}$ est définie comme la racine carrée de la somme des carrés de tous les éléments de la matrice :

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2}.$$

- **Expression avec la trace** : La norme de Frobenius peut également être exprimée à l'aide de la trace :

$$\|\mathbf{A}\|_F = \sqrt{\text{tr}(\mathbf{A}^{\text{T}} \mathbf{A})}.$$

- **Propriétés** :
  - $\|\mathbf{A}\|_F \geq 0$, et $\|\mathbf{A}\|_F = 0$ si et seulement si $\mathbf{A} = \mathbf{0}$.
  - $\|\alpha \mathbf{A}\|_F = |\alpha| \|\mathbf{A}\|_F$ pour $\alpha \in \mathbb{R}$.
  - $\|\mathbf{A} + \mathbf{B}\|_F \leq \|\mathbf{A}\|_F + \|\mathbf{B}\|_F$ (inégalité triangulaire).

Norme de Frobenius : Exemple Numérique

- Soit la matrice suivante :

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}.$$

La norme de Frobenius est donnée par :

$$\|\mathbf{A}\|_F = \sqrt{1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2} = \sqrt{91}.$$

- Avec la trace, on peut aussi calculer :

$$\|\mathbf{A}\|_F = \sqrt{\text{tr}(\mathbf{A}^{\mathrm{T}}\mathbf{A})}.$$

Norme de Frobenius : Intuition

- **Norme de Frobenius (pour les matrices)** :
  - **Interprétation** : La norme de Frobenius est analogue à la norme ℓ_2, mais pour les matrices.
  - **Application** : Souvent utilisée pour quantifier les erreurs ou les différences entre matrices, comme dans les problèmes de factorisation de matrices ou d'approximation.
    - **Exemple** : La norme de Frobenius peut évaluer l'écart entre une matrice estimée et une matrice réelle dans les problèmes de minimisation.

Norme de Frobenius : Code Python

```python
import numpy as np

# Définir la matrice
A = np.array([[1, 2, 3],
              [4, 5, 6]])

# Calculer la norme de Frobenius
frobenius_norm = np.linalg.norm(A, 'fro')

# Afficher le résultat
print("Matrice A :\n", A)
print("Norme de Frobenius : {:.4f}".format(frobenius_norm))
```

Résultat :

```
Matrice A :
[[1 2 3]
 [4 5 6]]
Norme de Frobenius : 9.5394
```

Indépendance Linéaire et Rang d'une Matrice

Indépendance Linéaire

- **Définition** :
  - Un ensemble de vecteurs $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^m$ est dit **(linéairement) indépendant** si aucun vecteur ne peut être représenté comme une combinaison linéaire des autres vecteurs.
  - Inversement, si un vecteur de l'ensemble peut être représenté comme une combinaison linéaire des autres vecteurs, alors les vecteurs sont dits **(linéairement) dépendants**.
  - Autrement dit, si :

$$\mathbf{x}_n = \sum_{i=1}^{n-1} \alpha_i \mathbf{x}_i$$

pour certains scalaires $\alpha_1, \ldots, \alpha_{n-1} \in \mathbb{R}$, alors les vecteurs $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ sont linéairement dépendants ; sinon, ils sont linéairement indépendants.

Indépendance Linéaire

- **Définition** : Soient $\mathbf{x}_1, \ldots, \mathbf{x}_n$ des vecteurs de l'espace vectoriel $\mathbb{R}^m$. L'ensemble $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ est dit :
  - **Linéairement indépendant** si la seule combinaison linéaire nulle,

$$\alpha_1 \mathbf{x}_1 + \cdots + \alpha_n \mathbf{x}_n = 0$$

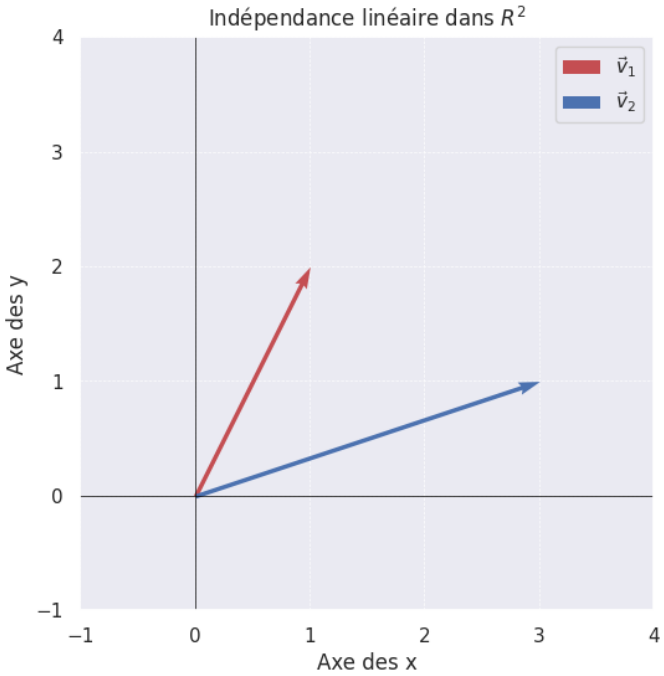
    est celle où $\alpha_1 = \alpha_2 = \cdots = \alpha_n = 0$.
  - **Linéairement dépendant** s'il existe des coefficients $\alpha_1, \alpha_2, \ldots, \lambda_n$, dont au moins un est non nul, tel que

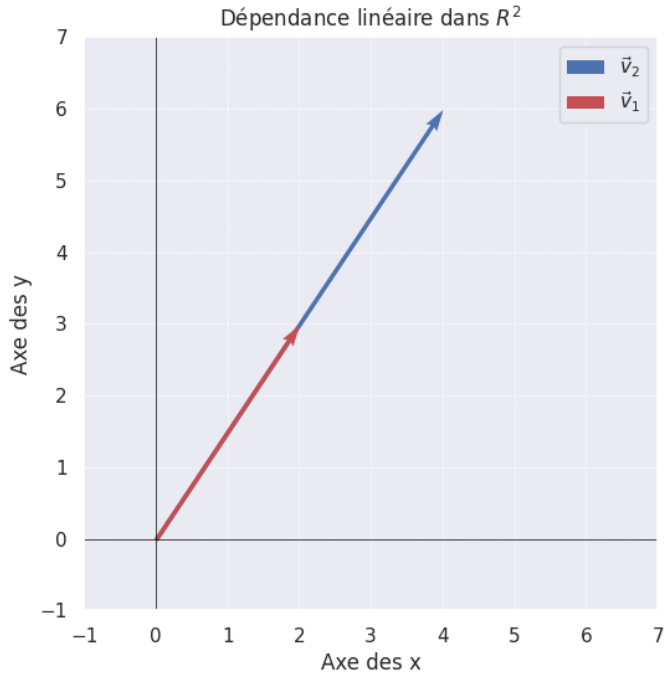
$$\alpha_1 \mathbf{x}_1 + \cdots + \alpha_n \mathbf{x}_n = 0.$$

- **Propriétés** : Soit $\mathbf{A} \in \mathbb{R}^{n \times n}$ (une matrice carrée).
  - Si $\det(\mathbf{A}) \neq 0$, les colonnes (ou lignes) de $\mathbf{A}$ sont **linéairement indépendantes**.
  - Si $\det(\mathbf{A}) = 0$, les colonnes (ou lignes) de $\mathbf{A}$ sont **linéairement dépendantes**.

Indépendance Linéaire : Interprétation Géométrique

- **Interprétation Géométrique :**
  - Dans $\mathbb{R}^2$: Des vecteurs linéairement indépendants ne sont pas sur la même droite.
  - Dans $\mathbb{R}^n$: Des vecteurs linéairement indépendants ne sont pas sur le même hyperplan.





Rang d'une Matrice

- **Définition :** Le **rang** d'une matrice A est le nombre maximal de lignes ou de colonnes linéairement indépendantes de A.
- **Propriétés :**
  - Pour $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\text{rang}(\mathbf{A}) \leq \min(m, n)$.
    Si $\text{rang}(\mathbf{A}) = \min(m, n)$, alors $\mathbf{A}$ est dite **de plein rang**.
  - Pour $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\text{rang}(\mathbf{A}) = \text{rang}(\mathbf{A}^{\mathrm{T}})$.
  - Pour $\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{n \times p}$, $\text{rang}(\mathbf{AB}) \leq \min(\text{rang}(\mathbf{A}), \text{rang}(\mathbf{B}))$.
  - Pour $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times n}$, $\text{rang}(\mathbf{A} + \mathbf{B}) \leq \text{rang}(\mathbf{A}) + \text{rang}(\mathbf{B})$.
- **Propriétés liées au Déterminant:**
  - Pour $\mathbf{A} \in \mathbb{R}^{n \times n}$, Si $\det(\mathbf{A}) \neq 0 \Rightarrow \text{rang}(\mathbf{A}) = n$ (càd, les colonnes et les lignes de $\mathbf{A}$ sont linéairement indépendantes).
  - Si $\det(\mathbf{A}) = 0 \Rightarrow \text{rang}(\mathbf{A}) < n$ (càd, la matrice possède des colonnes (ou lignes) linéairement dépendantes.

Indépendance Linéaire et Rang : Exemple Numérique

- **Exemple 1 : Dépendance Linéaire**
  Par exemple, les vecteurs :

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 4 \\ 1 \\ 5 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} 2 \\ -3 \\ -1 \end{bmatrix}$$

  sont linéairement dépendants car :

$$\mathbf{x}_3 = -2\mathbf{x}_1 + \mathbf{x}_2.$$

- **Exemple 2 : Indépendance Linéaire**
  Par contre, les vecteurs :

$$\mathbf{y}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{y}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{y}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

  - Aucune combinaison linéaire des deux premiers vecteurs ne peut donner le troisième vecteur.
  - **Le rang de la matrice formée par ces vecteurs est égal à** 3.

Indépendance Linéaire et Rang : Base d'un Espace Vect

- **Définition :** Soient $\mathbf{x}_1, \ldots, \mathbf{x}_n$ des vecteurs de l'espace vectoriel V. L'ensemble $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ constitue une **base** de l'espace vectoriel V si l'ensemble satisfait les propriétés suivantes :
  - $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ **engendre** V (tout vecteur de V peut être écrit comme une combinaison linéaire des vecteurs de la base).
  - $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ Est **linéairement indépendant**.
- **Propriétés clés :**
  - Tout vecteur de V peut être exprimé de manière unique comme une combinaison linéaire des vecteurs de la base.
  - Le nombre de vecteurs dans une base est la **dimension** de V.
    - Par ex., $\{\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3\}$ tel que :

$$\mathbf{y}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{y}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{y}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

est une base de $\mathbb{R}^3$.

Indépendance Linéaire et Rang : Cas d'Utilisation

- **Résolution de Systèmes Linéaires :**
  - Solution unique si $\text{rang}(\mathbf{A}) = \text{nombre de colonnes}$.
  - Solutions infinies ou aucune si $\text{rang}(\mathbf{A}) < \text{nombre de colonnes}$.
- **Compression de Données :**
  - L'ACP (Analyse en Composantes Principales) utilise le rang pour identifier les dimensions significatives.
- **Application en Statistiques :**
  - Identification des relations de dépendance entre variables explicatives dans un modèle de régression.

Indépendance Linéaire et Rang : Code Python

```python
import numpy as np

# Définir une matrice
A = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

# Calculer le rang
rang = np.linalg.matrix_rank(A)
print(f"Le rang de la matrice est : {rang}")

# Vérifier l'indépendance linéaire des colonnes
def verifier_independance_lineaire(matrice):
    rang = np.linalg.matrix_rank(matrice)
    return rang == min(matrice.shape[0], matrice.shape[1])

independance = verifier_independance_lineaire(A)
print(f"Les colonnes sont-elles linéairement indépendantes ? {independance}")
```

Résultats:

```
Le rang de la matrice est : 2
Les colonnes sont-elles linéairement indépendantes ? False
```

Inverse d'une Matrice

Inverse d'une Matrice

- **Définition :** L'inverse d'une matrice carrée $\mathbf{A}$ de dimension $n \times n$, notée $\mathbf{A}^{-1}$, est une matrice telle que :

$$\mathbf{A} \cdot \mathbf{A}^{-1} = \mathbf{A}^{-1} \cdot \mathbf{A} = \mathbf{I}_n$$

où $\mathbf{I}_n$ est la matrice identité de dimension n.

- **Conditions d'existence :**
  - Une matrice $\mathbf{A}$ possède une inverse si, et seulement si, elle est inversible.
  - Une matrice est inversible si son déterminant est non nul : $\det(\mathbf{A}) \neq 0$.

Inverse d'une Matrice

- **Calcul de l'inverse :**
  - Pour une matrice $\mathbf{A} \in \mathbb{R}^{2\times 2}$ tel que $\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, :

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}, \quad \text{où } \det(\mathbf{A}) = ad - bc.$$

  - Pour une matrice $n \times n$ (avec $n > 2$), on peut utiliser la méthode des cofacteurs:

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} \cdot \text{adj}(\mathbf{A}),$$

    où $\text{adj}(\mathbf{A})$ est la matrice adjointe avec:

$$\text{adj}(\mathbf{A}) = \mathbf{C}^{\mathrm{T}} \text{ tel que } \mathbf{C} \text{ est la matrice de cofacteurs de } \mathbf{A} :$$

$$C_{ij} = (-1)^{i+j} \cdot \det(\mathbf{A}_{\setminus i, \setminus j})$$

  - **Note :** Des méthodes de décomposition comme la décomposition LU ou QR sont utilisées par les logiciels numériques.

Inverse d'une Matrice

- **Propriétés:**
  - Si $\mathbf{A}$ inversible $\Rightarrow$ alors son inverse est unique.
  - Si $\mathbf{A}$ et $\mathbf{B}$ inversibles $\Rightarrow$ $\mathbf{A} \cdot \mathbf{B}$ est également inversible tel que:

$$(\mathbf{A} \cdot \mathbf{B})^{-1} = \mathbf{B}^{-1} \cdot \mathbf{A}^{-1}.$$

  - Si $\mathbf{A}$ est inversible $\Rightarrow$ $\mathbf{A}^{\mathrm{T}}$ est également inversible, avec :

$$(\mathbf{A}^{\top})^{-1} = (\mathbf{A}^{-1})^{\top}.$$

  - Si $\mathbf{A}$ inversible $\Rightarrow$ pour tout entier positif k, on a :

$$(\mathbf{A}^{k})^{-1} = (\mathbf{A}^{-1})^{k}.$$

  - Si $\mathbf{A}$ est inversible $\Rightarrow$ $(\mathbf{A}^{-1})^{-1} = \mathbf{A}$.
  - Si $\mathbf{A}$ et $\mathbf{B}$ 2 matrices et $\mathbf{A}$ inversible $\Rightarrow$ $\mathrm{tr}(\mathbf{A}^{-1}\mathbf{B}) = \mathrm{tr}(\mathbf{B}\mathbf{A}^{-1})$.
  - Si $\mathbf{A}$ est une matrice inversible $\Rightarrow$ $\det(\mathbf{A}^{-1}) = \frac{1}{\det(\mathbf{A})}$.

Inverse d'une Matrice

- **Propriétés:**
  - $\text{diag}(\mathbf{v})^{-1} = \text{diag}\left(\left[\frac{1}{v_1}, \frac{1}{v_2}, \ldots, \frac{1}{v_n}\right]^{\mathrm{T}}\right)$
  - Si $\mathbf{A}$ est de la forme bloc-diagonale : $\mathbf{A} = \begin{pmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{0} & \mathbf{C} \end{pmatrix}$, $\Rightarrow \mathbf{A}$ est inversible si et seulement si $\mathbf{B}$ et $\mathbf{C}$ sont inversibles, avec :

$$\mathbf{A}^{-1} = \begin{pmatrix} \mathbf{B}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}^{-1} \end{pmatrix}.$$

- **Interprétation Géométrique et Intuition:**
  - $\mathbf{A}^{-1}$, représente une transformation qui "annule" l'effet de la transformation linéaire représentée par $\mathbf{A}$.
  - Si $\mathbf{A}$ transforme un espace en un autre, $\mathbf{A}^{-1}$ nous permet de revenir à l'espace initial.
  - Les lignes de $\mathbf{A}^{-1}$ peuvent être interprétées comme les vecteurs de base de l'espace d'origine exprimés en termes de l'espace transformé.

Inverse d'une matrice : Exemple Numérique

- **Problème :** Résolution de système linéaire :

$$\mathbf{Ax} = \mathbf{b} \quad \text{où} \quad \mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 5 \\ 7 \end{bmatrix}.$$

- **Trouver l'inverse de $\mathbf{A}$:**

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$

$$\det(\mathbf{A}) = (2)(3) - (1)(1) = 6 - 1 = 5,$$

$$\mathbf{A}^{-1} = \frac{1}{5} \begin{bmatrix} 3 & -1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 0.6 & -0.2 \\ -0.2 & 0.4 \end{bmatrix}.$$

- **Calcul de $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$:**

$$\mathbf{x} = \begin{bmatrix} 0.6 & -0.2 \\ -0.2 & 0.4 \end{bmatrix} \begin{bmatrix} 5 \\ 7 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}.$$

Inverse d'une Matrice : Code python

```python
import numpy as np

# Définir la matrice A et le vecteur b
A = np.array([[2, 1],
              [1, 3]])
b = np.array([5, 7])

# Calculer l'inverse de A
A_inv = np.linalg.inv(A)

# Résoudre pour x
x = np.dot(A_inv, b)

# Afficher le résultat
print("Vecteur solution x:", x)
```

Résultat :

```
Vecteur solution x: [3. 2.]
```

Orthogonalité d'une Matrice

Orthogonalité d'une Matrice

- **Définition :** Une matrice carrée $\mathbf{Q} \in \mathbb{R}^{n \times n}$ est dite orthogonale si elle satisfait :

$$\mathbf{Q}^{\mathrm{T}}\mathbf{Q} = \mathbf{Q}\mathbf{Q}^{\mathrm{T}} = \mathbf{I}_n,$$

- **Propriétés :**
  - Les colonnes (et les lignes) d'une matrice orthogonale $\mathbf{Q}$ sont orthonormées :

$$\mathbf{q}_i^{\mathrm{T}}\mathbf{q}_j = \begin{cases} 1, & \text{si } i = j \\ 0, & \text{si } i \neq j \end{cases}$$

  - Une orthogonale préserve la norme euclidienne des vecteurs :

$$\|\mathbf{Q}\mathbf{v}\|_2 = \|\mathbf{v}\|_2.$$

  - $\det(\mathbf{Q}) = \pm 1$.
    - Les matrices orthogonales propres ($\det(\mathbf{Q}) = 1$) représentent des **rotations**.
    - Les matrices orthogonales impropres ($\det(\mathbf{Q}) = -1$) combinent une **réflexion avec une rotation éventuelle**.
  - L'inverse d'une orthogonale est sa transposée : $\mathbf{Q}^{-1} = \mathbf{Q}^{\mathrm{T}}$.

Orthogonalité d'une Matrice

- **Exemples :**
  - La matrice identité $\mathbf{I}_n$ est orthogonale.
  - Une matrice de rotation en 2D :

$$\mathbf{Q} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

    est orthogonale.
  - La matrice de permutation des axes de coordonnées est orthogonale.

$$\text{Par exemple, dans } \mathbb{R}^4, \quad \mathbf{P} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

- **Applications :**
  - Réduction dimensionnelle (ex. : décomposition en valeurs singulières, PCA) utilisées en compression des données.
  - Transformations dans les espaces euclidiens (ex. : rotations, changements de base) utilisées beaucoup dans les application graphiques.

Orthogonalité d'une Matrice : Interprétation Géométrique

- **Distances :** Pour tout vecteur $\mathbf{v} \in \mathbb{R}^n$, la transformation par une matrice orthogonale $\mathbf{Q}$ satisfait :

$$\|\mathbf{Q}\mathbf{v}\|_2 = \|\mathbf{v}\|_2.$$

  Cela signifie que la longueur des vecteurs reste inchangée.
- **Angles :** Le produit scalaire est préservé :

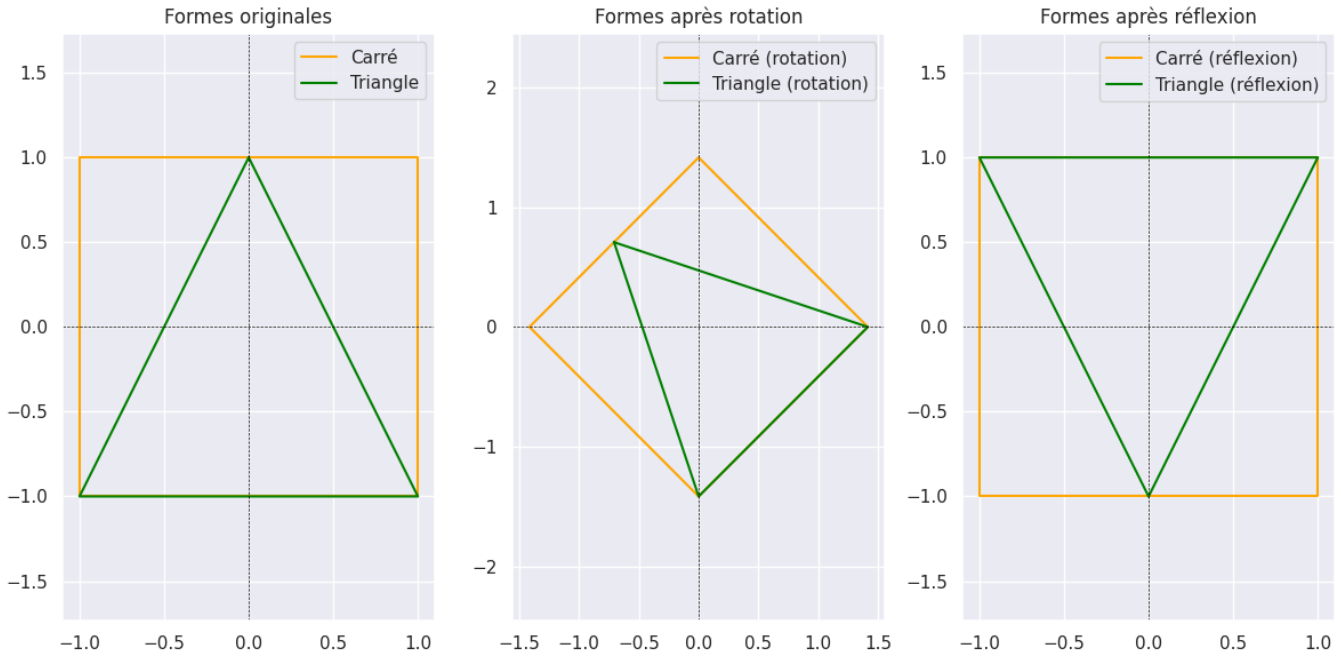
$$(\mathbf{Q}\mathbf{u})^{\mathrm{T}}(\mathbf{Q}\mathbf{v}) = \mathbf{u}^{\mathrm{T}}\mathbf{Q}^{\mathrm{T}}\mathbf{Q}\mathbf{v} = \mathbf{u}^{\mathrm{T}}\mathbf{I}_n\mathbf{v} = \mathbf{u}^{\mathrm{T}}\mathbf{v}.$$

  Comme le produit scalaire définit le cosinus de l'angle entre les vecteurs, les angles entre les vecteurs sont conservés :

$$\cos\theta = \frac{\mathbf{u}^{\mathrm{T}}\mathbf{v}}{\|\mathbf{u}\|_2\|\mathbf{v}\|_2}.$$

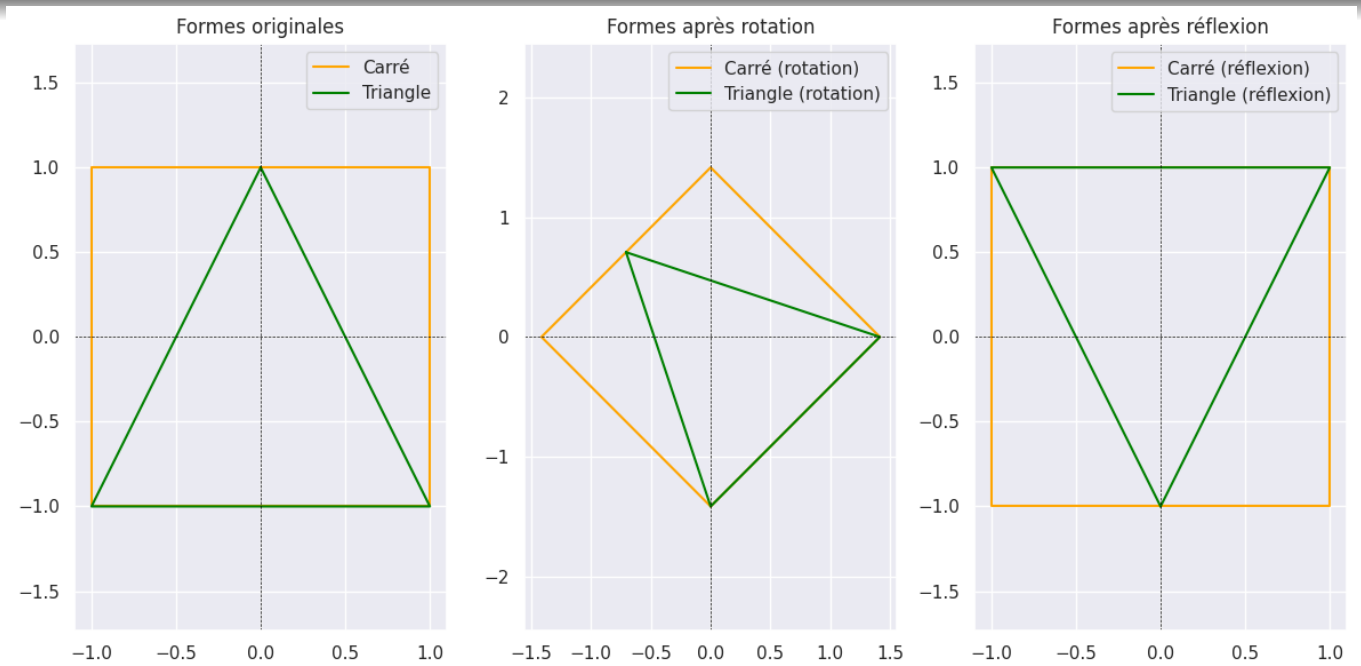
- **Préservation des Formes :** Les formes dans $\mathbb{R}^n$ ne sont pas déformées. Une sphère reste une sphère, et un cube reste un cube, bien que leur orientation puisse changer.

Orthogonalité d'une Matrice : Interprétation Géométrique



- **Effet de la rotation :** $\mathbf{R} = \begin{bmatrix} \cos(\pi/4) & -\sin(\pi/4) \\ \sin(\pi/4) & \cos(\pi/4) \end{bmatrix}$ Les formes (carré et triangle) conservent leur géométrie (angles et distances) mais changent d'orientation en raison de la rotation de $45°$.

Orthogonalité d'une Matrice : Interprétation Géométrique



- **Effet de la réflexion :**

$$\mathbf{M} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Effet de la réflexion : Les formes (carré et triangle) sont reflétées par rapport à l'axe y, préservant leur géométrie tout en inversant leur orientation.

Orthogonalité d'une Matrice : Exemple Numérique

- Considérons une matrice de rotation en 2D avec un angle de $\theta = \pi/4$ $(45°)$:

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 0.707 & -0.707 \\ 0.707 & 0.707 \end{bmatrix}.$$

- Vérifions les propriétés d'orthogonalité :
  - **Produit avec sa transposée :** $\mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{I}_2$, donc la matrice est orthogonale.
  - **Préservation de la norme :** Pour un vecteur $\mathbf{v} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, la norme du vecteur transformé reste inchangée :

$$\|\mathbf{R}\mathbf{v}\|_2 = \|\mathbf{v}\|_2 = 1.$$

  - **Vérifions cela avec Python.**

Orthogonalité d'une Matrice : Code Python

```python
import numpy as np
# Exemple : Matrice de rotation en 2D
theta = np.pi / 4  # Angle de 45 degrés
R = np.array([[np.cos(theta), -np.sin(theta)],
              [np.sin(theta), np.cos(theta)]])
# Vérification de l'orthogonalité
# R.T @ R est equivalent à np.dot(R.T, R)
orthogonal_check = np.allclose(R.T @ R, np.eye(2))
print("Matrice R :")
print(R)
print(f"La matrice R est-elle orthogonale ? {'Oui' if orthogonal_check else 'Non'}")
# Norme préservée
v = np.array([1, 0])
transformed_v = R @ v
original_norm = np.linalg.norm(v)
transformed_norm = np.linalg.norm(transformed_v)
print(f"Norme du vecteur original : {original_norm:.2f}")
print(f"Norme du vecteur transformé : {transformed_norm:.2f}")
```

Résultat :

```
Matrice R :
[[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]
La matrice R est-elle orthogonale ? Oui
Norme du vecteur original : 1.00
Norme du vecteur transformé : 1.00
```

Valeurs et Vecteurs Propres

Valeurs et Vecteurs Propres

- **Définition :** Pour une matrice carrée $\mathbf{A} \in \mathbb{R}^{n \times n}$, un scalaire $\lambda \in \mathbb{R}$ est appelé une **valeur propre** et un vecteur non nul $\mathbf{v} \in \mathbb{R}^n$ est appelé un **vecteur propre** associé à λ, s'ils satisfont :

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{v}.$$

- **Interprétation Géométrique :**
  - Le vecteur propre $\mathbf{v}$ représente une direction qui reste inchangée (mais peut être mise à l'échelle) par la transformation $\mathbf{A}$.
  - La valeur propre λ représente le facteur d'échelle dans cette direction.

Valeurs et Vecteurs Propres

- **Valeurs propres :** Résoudre l'**équation caractéristique** :

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

où $\mathbf{I}$ est la matrice identité. Les racines de ce polynôme sont les valeurs propres.

- **Vecteurs propres :** Pour chaque valeur propre λ, résoudre :

$$(\mathbf{A} - \lambda \mathbf{I})\mathbf{v} = \mathbf{0}.$$

Ce système d'équations donne les vecteurs propres associés à λ.

- **Échelle :** La valeur propre λ indique comment le vecteur propre est mis à l'échelle :
  - $\lambda > 1$: étirement.
  - $0 < \lambda < 1$: compression.
  - $\lambda = 1$: aucune mise à l'échelle.
  - $\lambda = 0$: écrasement au vecteur nul.
  - $\lambda < 0$: réflexion et mise à l'échelle.

Valeurs et Vecteurs Propres : Exemples

Matrices de Réflexion :

- Exemple : Une matrice avec ses valeurs et vecteurs propres :

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \lambda_1 = 1, \quad \mathbf{v}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \lambda_2 = -1.$$

Matrices de Rotation :

- Exemple : La matrice

$$\mathbf{A} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

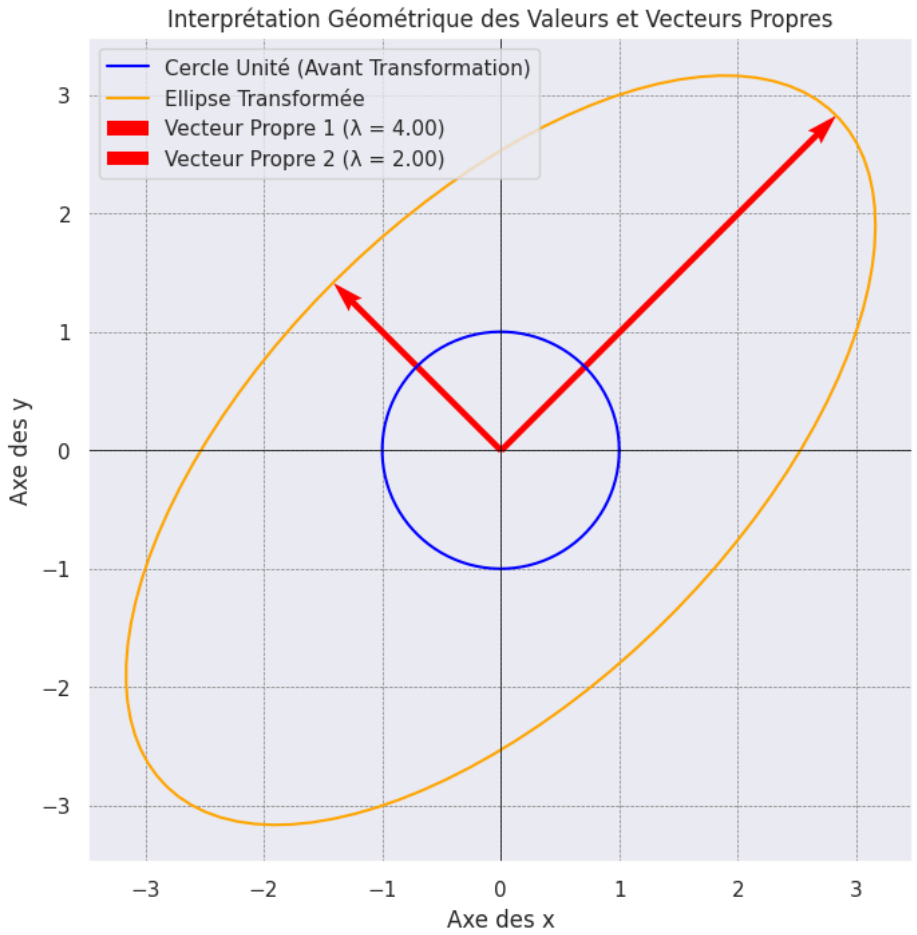
- Valeurs propres complexes :

$$\lambda = e^{i\theta}, \quad \lambda = e^{-i\theta}.$$

- Interprétation : Cette matrice effectue une rotation des vecteurs d'un angle θ dans $\mathbb{R}^2$.

Valeurs et Vecteurs Propres : Interprétation Géométrique

- **Matrice de Transformation, valeurs et vecteurs propres:**
  $\mathbf{A} = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$, $\mathbf{v}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $\mathbf{v}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$, $\lambda_1 = 4$, $\lambda_2 = 2$.
- **Interprétation Géométrique :**
  - Le cercle unité (en bleu) est transformé en une ellipse (en orange) par $\mathbf{A}$.
  - Les vecteurs propres (en rouge) indiquent les directions invariantes.
  - les valeurs propres (λ) déterminent l'échelle le long de ces directions.



Propriétés des Valeurs et Vecteurs Propres

- La somme des valeurs propres est égale à la trace de la matrice :

$$\text{tr}(\mathbf{A}) = \sum_{i=1}^{n} \lambda_i.$$

- Le produit des valeurs propres est égal au déterminant de la matrice :

$$\det(\mathbf{A}) = \prod_{i=1}^{n} \lambda_i.$$

- **Valeurs Propres Nulles :** Si $\lambda = 0$ est une valeur propre, alors la matrice en question est singulière (non inversible).
- **Indépendance Linéaire :** Les vecteurs propres associés à des valeurs propres distinctes sont linéairement indépendants.
- **Pour une matrice symétrique** $(\mathbf{A} = \mathbf{A}^T)$:
  - Les vecteurs propres associés à des valeurs propres distinctes sont orthogonaux.
  - Les valeurs propres sont toujours réelles.

Valeurs et Vecteurs Propres : Propriétés

- **Pour une matrice anti-symétrique** ($\mathbf{A} = -\mathbf{A}^T$), les valeurs propres sont purement imaginaires ou nulles.
- **Pour une matrice diagonale** $\mathbf{D} = \text{diag}(\mathbf{v})$, les valeurs propres sont les diagonaux (les éléments de $\mathbf{v}$) et les vecteurs propres sont les vecteurs unitaires de la base standard.
- **pour la matrice Identité** $\mathbf{I}_n$ Les valeurs propres sont les éléments diagonaux et tout vecteur de $\mathbb{R}^n$ non nul est un vecteur propre.
- **Pour une matrice orthogonale** ($\mathbf{A}^{\mathrm{T}}\mathbf{A} = \mathbf{A}\mathbf{A}^{\mathrm{T}} = \mathbf{I}$), Les valeurs propres ont une valeur absolue égale à 1 ($\lambda = \pm 1$) ou des valeurs complexes sur le cercle unité ($|\lambda| = 1$). Les vecteurs propres associés à des valeurs propres distinctes sont orthogonaux.

Valeurs et Vecteurs Propres : Cas d'Utilisation

- **Analyse en Composantes Principales (ACP) :**
  - Les vecteurs propres représentent les directions principales de la variance des données.
  - Les valeurs propres mesurent la variance le long de ces directions.
- **Systèmes Dynamiques :**
  - Les valeurs propres déterminent la stabilité des points d'équilibre.
  - Valeurs propres positives : croissance, négatives : décroissance.
- **Mécanique Quantique :**
  - Les valeurs propres représentent des quantités mesurables, comme les niveaux d'énergie.
  - Les vecteurs propres représentent les états correspondants.
- **Élasticité et Déformation :**
  - Les valeurs propres décrivent les contraintes principales (stress) dans les matériaux.
  - Les vecteurs propres indiquent les directions principales associées.

Valeurs et Vecteurs Propres : Code Python

```python
import numpy as np
from numpy.linalg import eig
# Exemple : Définir une matrice
A = np.array([[3, 1],
              [1, 3]])
# Calculer les valeurs propres et les vecteurs propres
eigenvalues, eigenvectors = eig(A)
print("Matrice A:")
print(A)
# Valeurs propres
print("Valeurs propres:")
print(eigenvalues)
# Vecteurs propres
print("Vecteurs propres:")
print(eigenvectors)
```

Résultats :

```
Matrice A:
[[3 1]
 [1 3]]
Valeurs propres:
[4. 2.]
Vecteurs propres:
[[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]
```

Valeurs et Vecteurs Propres : Code Python

```python
# Vérification : A * v = lambda * v pour chaque vecteur propre
for i in range(len(eigenvalues)):
    v = eigenvectors[:, i]
    lambda_v = eigenvalues[i] * v
    Av = np.dot(A, v)
    print(f"Vérification pour lambda = {eigenvalues[i]:.2f}:")
    print("A * v =", Av)
    print("lambda * v =", lambda_v)
    print("Égalité :", np.allclose(Av, lambda_v))
```

Résultats :

```
Vérification pour lambda = 4.00:
A * v = [2.82842712 2.82842712]
lambda * v = [2.82842712 2.82842712]
Égalité : True
Vérification pour lambda = 2.00:
A * v = [-1.41421356  1.41421356]
lambda * v = [-1.41421356  1.41421356]
Égalité : True
```

Décomposition en Éléments Propres

Décomposition en Éléments Propres

Définition :

- La décomposition en valeurs propres d'une matrice carrée $\mathbf{A} \in \mathbb{R}^{n \times n}$ consiste à exprimer $\mathbf{A}$ comme :

$$\mathbf{A} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^{-1},$$

où :

  - $\mathbf{\Lambda} = \text{diag}(\boldsymbol{\lambda})$ est la matrice diagonale des valeurs propres de $\mathbf{A}$.
  - $\mathbf{P}$ est une matrice dont les colonnes sont les vecteurs propres de $\mathbf{A}$.

- Cela est possible si $\mathbf{A}$ possède n vecteurs propres linéairement indépendants.

Décomposition en Éléments Propres

- **Calcul de l'inverse :**

$$\mathbf{A}^{-1} = \mathbf{P}\mathbf{\Lambda}^{-1}\mathbf{P}^{-1},$$

où $\mathbf{\Lambda}$ est la matrice diagonale contenant les valeurs propres de $\mathbf{A}$ et $\mathbf{\Lambda}^{-1}$ est obtenue en inversant chaque valeur propre sur la diagonale de $\mathbf{\Lambda}$.

- **Calcul des Puissances :**

$$\mathbf{A}^{k} = \mathbf{P}\mathbf{\Lambda}^{k}\mathbf{P}^{-1},$$

où $\mathbf{\Lambda}^{k}$ est la matrice diagonale avec les valeurs propres élevées à la puissance k.

- **Exponentielle de Matrice :**

$$e^{\mathbf{A}} = \mathbf{P}e^{\mathbf{\Lambda}}\mathbf{P}^{-1},$$

où $e^{\mathbf{\Lambda}}$ applique l'exponentielle à chaque valeur propre.

Décomposition en Éléments Propres

Propriétés des Matrices Symétriques :

- $\mathbf{A} = \mathbf{A}^{\mathrm{T}}$ (matrice symétrique).
- La décomposition est toujours possible pour les matrices symétriques puisque les vecteurs propres sont orthogonaux $\mathbf{P}^{-1} = \mathbf{P}^{\mathrm{T}}$.
- Décomposition en valeurs propres :

$$\mathbf{A} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^{\mathrm{T}},$$

où :

- $\mathbf{\Lambda}$ est diagonale avec les valeurs propres réelles.
- $\mathbf{Q}$ est orthogonale ($\mathbf{P}^{\mathrm{T}}\mathbf{P} = \mathbf{P}\mathbf{P}^{\mathrm{T}} = \mathbf{I}$).

Décomposition en Éléments Propres

Propriétés des Matrices Orthogonales :

- $\mathbf{Q}^{\mathrm{T}}\mathbf{Q} = \mathbf{I}$ (matrice orthogonale).
- La décomposition est toujours possible pour les matrices orthogonales puisque les vecteurs propres toujours distincts et orthogonaux
- Décomposition en valeurs propres :

$$\mathbf{Q} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^{\mathrm{T}},$$

où :

- $\mathbf{\Lambda}$ contient les valeurs propres $|\lambda_i| = 1$.
- $\mathbf{P}$ est une matrice orthogonale.

Décomposition en Éléments Propres

Exemple d'une Matrice Orthogonale :

$$\mathbf{Q} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}.$$

Décomposition en Valeurs Propres :

- **Valeurs Propres :**

$$\mathbf{Q} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^{\mathrm{T}}, \text{ où tel que } \lambda_1 = i,\ \lambda_2 = -i \text{ et } \mathbf{P} = \begin{bmatrix} 1 & 1 \\ -i & i \end{bmatrix}.$$

Interprétation :

- $\mathbf{Q}$ représente une rotation de $90°$ dans le sens anti-horaire dans $\mathbb{R}^2$.
- Les valeurs propres $\lambda_1 = i$ et $\lambda_2 = -i$ indiquent une transformation complexe sur le cercle unité.
- Les vecteurs propres associés définissent des directions invariantes sous la transformation.

Décomposition en Élément Propres : Code Python

```python
import numpy as np
# Définir la matrice
A = np.array([[4, 1],
              [1, 4]])

# Décomposition en valeurs propres
eigenvalues, eigenvectors = np.linalg.eig(A)

# Matrice diagonale des valeurs propres
Lambda = np.diag(eigenvalues)

# Vérifier la décomposition
P = eigenvectors
P_inv = np.linalg.inv(P)
A_reconstructed = P @ Lambda @ P_inv
```

Décomposition en Élément Propres : Code Python

```python
# Afficher les résultats
print("Matrice originale A:", A)
print("Valeurs propres:", eigenvalues)
print("Vecteurs propres (colonnes de P):", P)
print("Matrice diagonale Lambda (valeurs propres):", Lambda)
print("Matrice inverse de P:", P_inv)
print("Matrice reconstruite (A_reconstructed):", A_reconstructed)
```

Résultats :

```
Matrice originale A: [[4 1]
 [1 4]]
Valeurs propres: [5. 3.]
Vecteurs propres (colonnes de P): [[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]
Matrice diagonale Lambda (valeurs propres): [[5. 0.]
 [0. 3.]]
Matrice inverse de P: [[ 0.70710678  0.70710678]
 [-0.70710678  0.70710678]]
Matrice reconstruite (A_reconstructed): [[4. 1.]
 [1. 4.]]
```

Formes Quadratiques et Matrices Semi-Définies Positives

Formes Quadratiques et Matrices SDP

- Une matrice $\mathbf{A}$ est **semi-définie positive** (SDP) si et seulement si :

$$q(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x} = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j \geq 0 \quad \forall\, \mathbf{x} \in \mathbb{R}^n.$$

- Une matrice $\mathbf{A}$ est **semi-définie positive** si toutes les valeurs propres de $\mathbf{A}$ sont positives :

$$\lambda_i \geq 0 \quad \text{pour tout } i.$$

- Les valeurs propres de $\mathbf{A}$ déterminent la forme de la surface.
- Les vecteurs propres de $\mathbf{A}$ représentent les directions principales (axes).

Formes Quadratiques et Matrices SDP

Pour une matrice $\mathbf{A} \in \mathbb{R}^{2\times 2}$, la forme quadratique :

$$q(x_1, x_2) = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

décrit une surface dans $\mathbb{R}^3$.

Cas principaux :

- Matrice SDP : paraboloïde elliptique (valeurs propres ≥ 0).
- Matrice indéfinie : surface en forme de selle (valeurs propres mixtes).
- Matrice SDN : paraboloïde elliptique inversé (valeurs propres ≤ 0).

Formes Quadratiques et SDP : Exemples Numériques

Exemple 1 : Matrice SDP

$$\mathbf{A} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}.$$

- Valeurs propres : $\lambda_1 = 3, \lambda_2 = 1$ $(\lambda_i \geq 0)$.
- Forme quadratique : $q(\mathbf{x}) = 2x_1^2 - 2x_1x_2 + 2x_2^2 \geq 0$.

Exemple 2 : Matrice Indéfinie

$$\mathbf{A} = \begin{bmatrix} 2 & -3 \\ -3 & 2 \end{bmatrix}.$$

- Valeurs propres : $\lambda_1 = 5, \lambda_2 = -1$ (signes mixtes).
- Forme quadratique : $q(\mathbf{x}) = 2x_1^2 - 6x_1x_2 + 2x_2^2$.
- Surface : Forme de selle.

Formes Quadratiques et SDP : Interprétation Géométrique

- Les formes quadratiques définissent la courbure des surfaces :
  - **Matrice SDP :** Paraboloïde elliptique ouvert vers le haut.
  - **Matrice SDN :** Paraboloïde elliptique ouvert vers le bas.
  - **Matrice Indéfinie :** Surface en forme de selle.

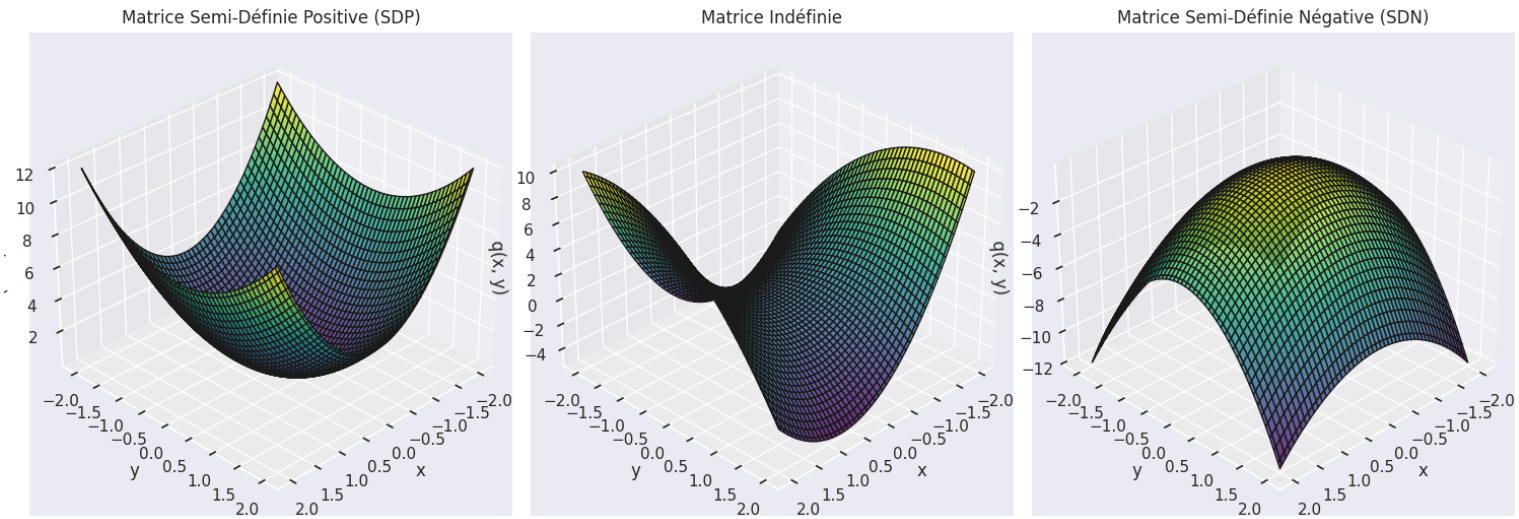


Figure: Interprétation géométrique des formes quadratiques.

Formes Quadratiques et SDP : Code Python

```python
import numpy as np

# Définir la matrice A et le vecteur x
A = np.array([[2, 1],
              [1, 3]])
x = np.array([1, 2])

# Calcul de la forme quadratique
q = x.T @ A @ x

print("Matrice A:")
print(A)
print("Vecteur x:")
print(x)
print("Valeur de la forme quadratique q(x):", q)
```

Résultat :

```
Matrice A:
[[2 1]
 [1 3]]
Vecteur x:
[1 2]
Valeur de la forme quadratique q(x): 12
```

Formes Quadratiques et SDP : Cas d'Utilisation

- **Optimisation :**
  - $q(\mathbf{x})$ est centrale dans les problèmes d'optimisation, car elle décrit souvent la fonction objective (par exemple, dans la programmation quadratique)
- **Physique et ingénierie :**
  - Décrire l'énergie et la stabilité des systèmes physiques.
- **Apprentissage automatique :**
  - Utilisées dans les noyaux pour les SVM et l'ACP.
- **Systèmes dynamiques :**
  - Les fonctions de Lyapunov basées sur les matrices SDP garantissent la stabilité.

Calcul Matriciel

Le Gradient et la Matrice Jacobienne

Gradient et Jacobienne

- **Gradient :** Le gradient d'une fonction scalaire $f : \mathbb{R}^n \to \mathbb{R}$ par rapport au vecteur $\mathbf{x} = \left[x_1, x_2, \cdots, x_n\right]^{\mathrm{T}}$ est :

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}.$$

- Le gradient pointe dans la direction de la plus grande augmentation de f.
- **Jacobienne :** : Pour une fonction $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, la matrice Jacobienne est définie comme :

$$\mathbf{J_f}(\mathbf{x}) = \left[\frac{\partial \mathbf{f}}{\partial x_1}, \frac{\partial \mathbf{f}}{\partial x_2}, \cdots, \frac{\partial \mathbf{f}}{\partial x_n}\right] = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}.$$

- La Jacobienne généralise le gradient pour les fonctions vectorielles.

Le Gradient d'une Fonction à Plusieurs Variables

Gradient : Exemple Numérique

- Le gradient d'une fonction scalaire $f : \mathbb{R}^n \to \mathbb{R}$ par rapport au vecteur $\mathbf{x} = \left[x_1, x_2, \cdots, x_n\right]^{\mathrm{T}}$ est :

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}\right]^{\mathrm{T}}.$$

- **Pour l'exemple du gradient,** Considérons la fonction scalaire :

$$f(x_1, x_2) = -x_1^2 - 3x_1x_2 - 4x_2^2.$$

- Son gradient est donné par :

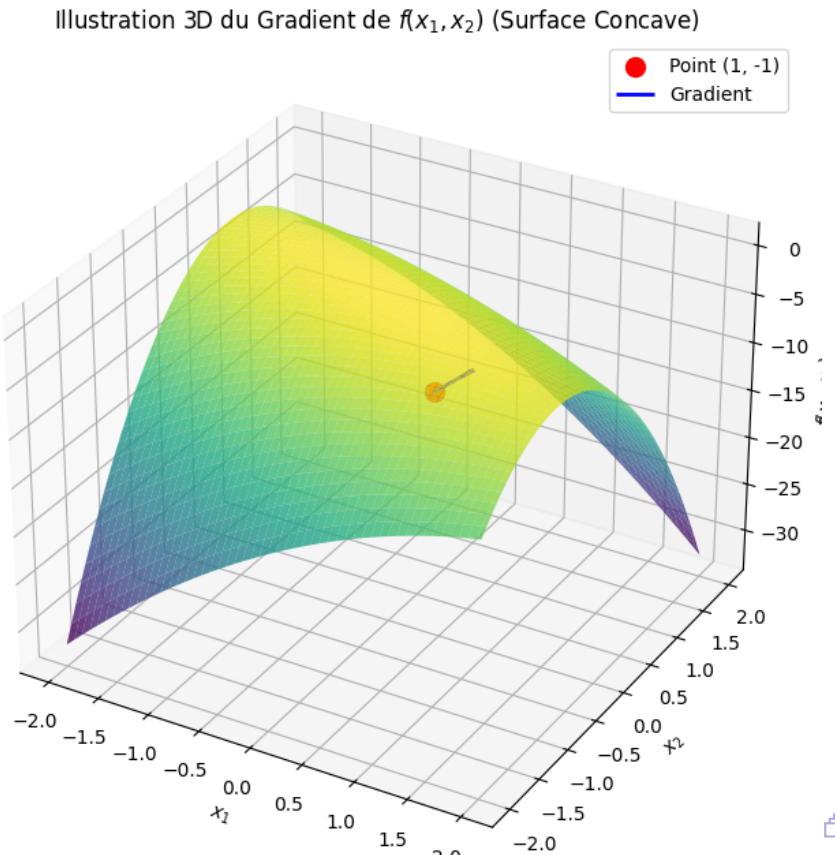
$$\nabla f(\mathbf{x}) = \begin{bmatrix} -2x_1 - 3x_2 \\ -3x_1 - 8x_2 \end{bmatrix}.$$

- En $(x_1, x_2) = (1, -1)$, le gradient est :

$$\nabla f(1, -1) = \begin{bmatrix} 1 \\ 5 \end{bmatrix}.$$

Gradient : Interprétation Géometrique

- La figure ci-dessous illustre la surface concave définie par $f(x_1, x_2)$, ainsi que le gradient ∇f en $(x_1, x_2) = (1, -1)$.
- Le gradient (vecteur bleu) pointe dans la direction de la plus grande augmentation de f.



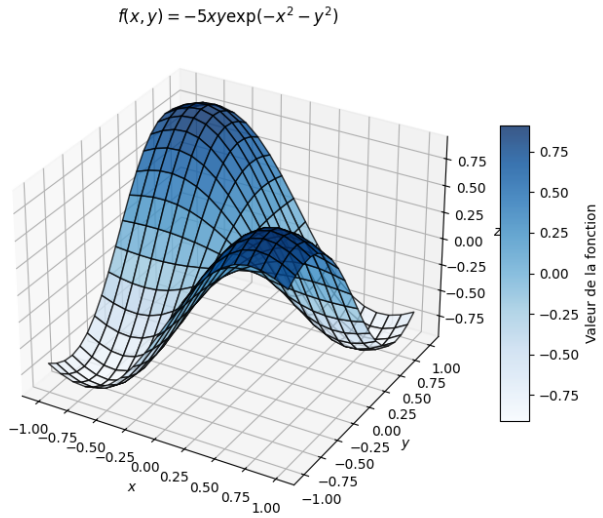
Gradient : Interprétation Géometrique

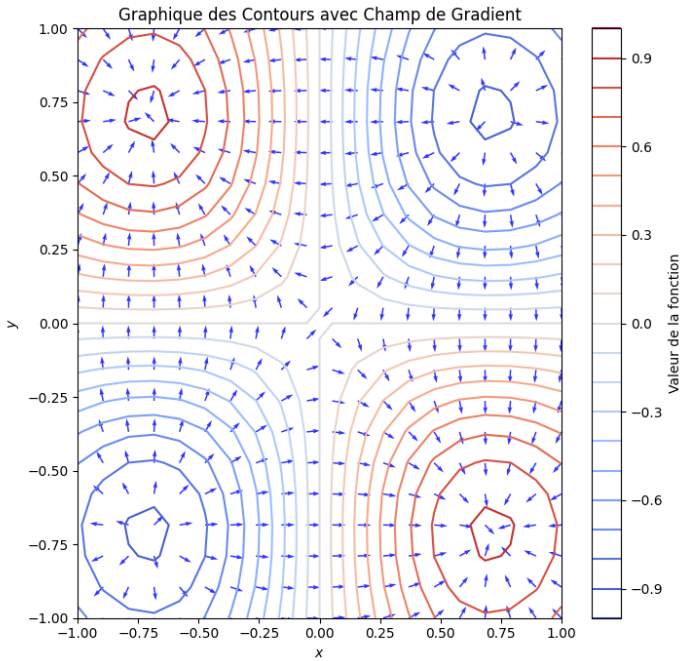
- **Pour des fins de visualisation,** considérons la fonction :

$$f(x_1, x_2) = -5x_1x_2 \exp(-x_1^2 - x_2^2)$$

- **Expression du gradient :**

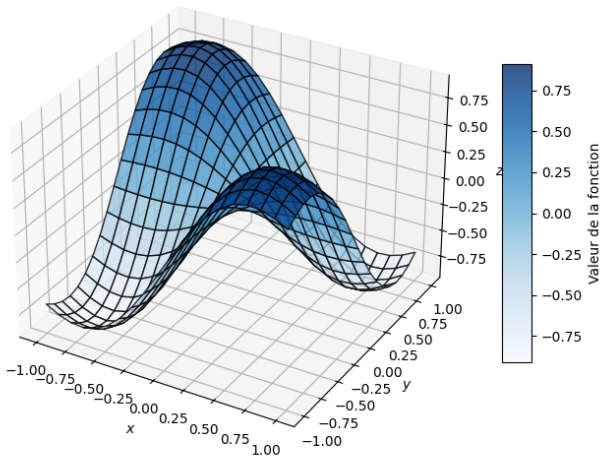
$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -5x_2(1 - 2x_1^2) \exp(-x_1^2 - x_2^2) \\ -5x_1(1 - 2x_2^2) \exp(-x_1^2 - x_2^2) \end{bmatrix}.$$

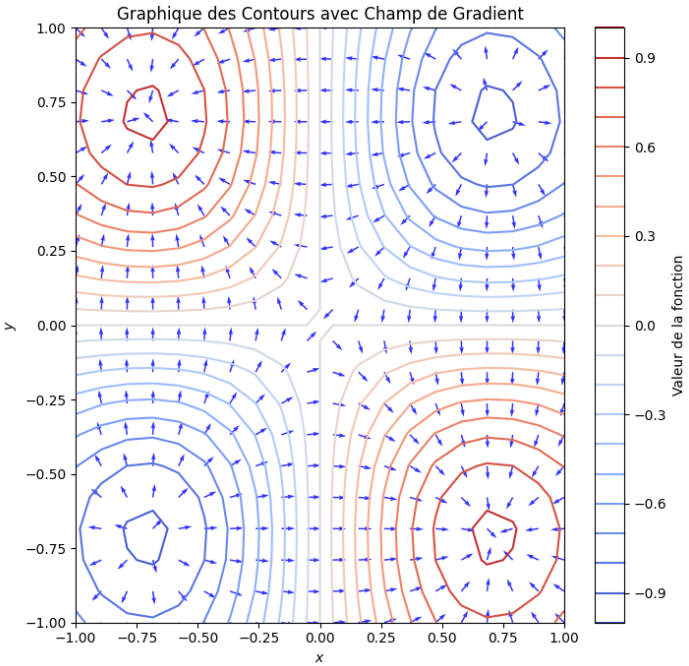




Gradient : Interprétation Géometrique

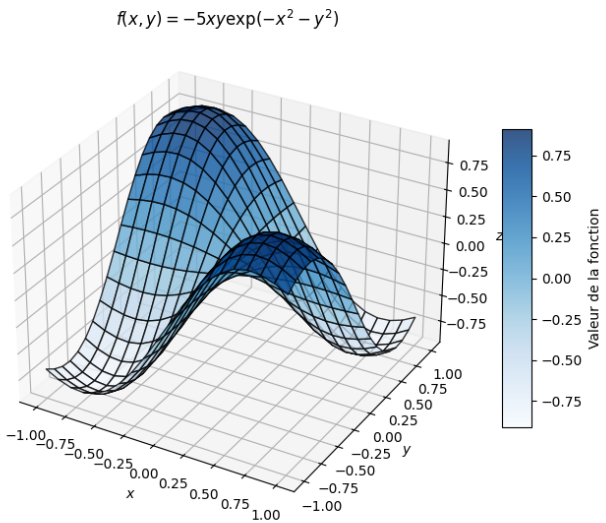
- Le gradient $\nabla f(x_1, x_2)$ pointe dans la direction de l'augmentation la plus rapide de la fonction scalaire $f(x_1, x_2)$ au point donné.
- Inversement, $-\nabla f(x_1, x_2)$ pointe dans la direction de la plus grande diminution.

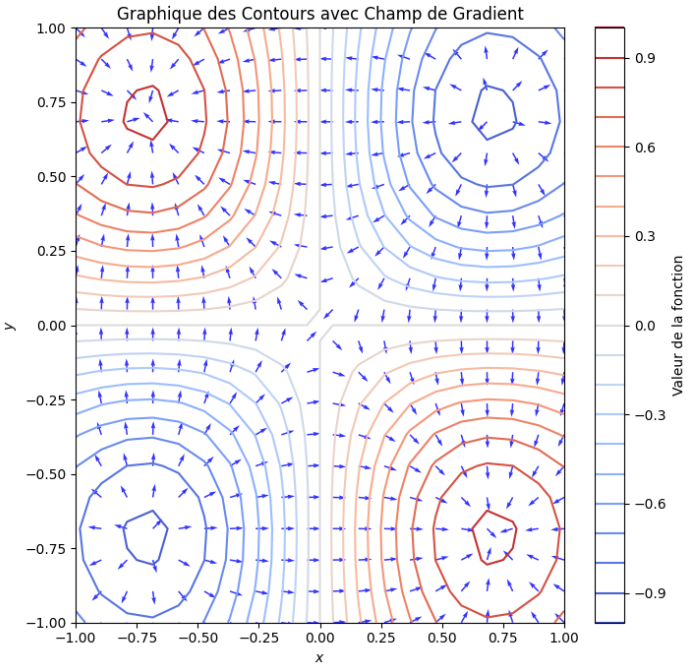




Gradient : Interprétation Géometrique

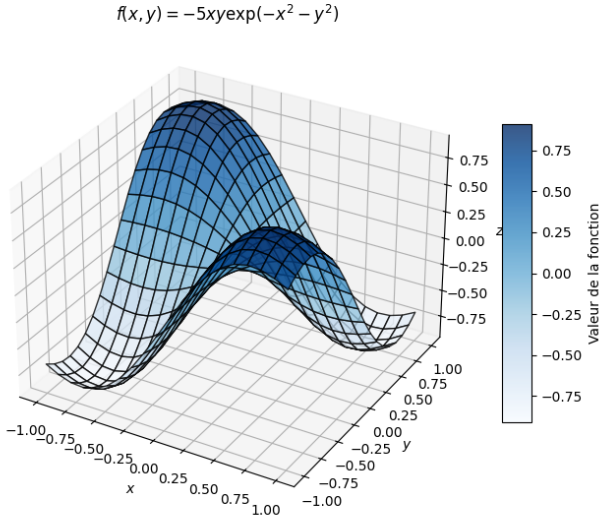
- Le gradient est toujours **orthogonal** (perpendiculaire) aux courbes de niveau (ou contours) de $f(x_1, x_2)$.
- Si l'on trace les contours de $f(x_1, x_2)$, les vecteurs du gradient intersecteront ces courbes à angle droit.

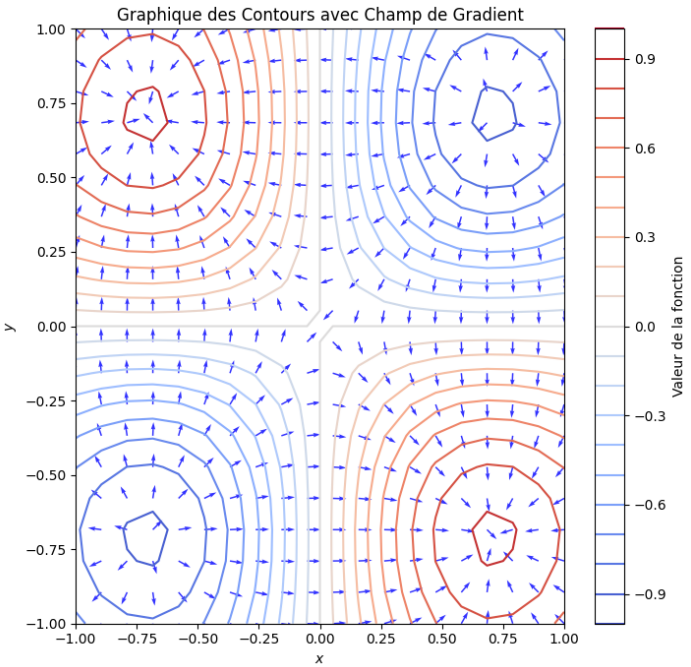




Gradient : Interprétation Géometrique

- La magnitude $\|\nabla f(x_1, x_2)\|$ représente la rapidité du changement de f dans la direction du gradient.
- Une grande magnitude indique une pente raide.
- Une petite magnitude implique une surface plus plate.





Gradient : Code Python

```python
from sympy import symbols, diff

# 1. Define symbolic variables
x, y = symbols('x y')

# 2. Define the scalar function
f = -x**2 - 3*x*y - 4*y**2

# 3. Compute the gradient symbolically
grad_f = [diff(f, var) for var in (x, y)]

# Display the gradient
print(f"Gradient: {grad_f}")

# 4. Evaluate the gradient at (1, -1)
point = {x: 1, y: -1}
evaluated_grad = [grad.subs(point) for grad in grad_f]
print(f"Gradient at {point}: {evaluated_grad}")
```

Résultat :

```
Gradient: [-2*x - 3*y, -3*x - 8*y]
Gradient at {x: 1, y: -1}: [1, 5]
```

La Matrice Jacobienne

La Matrice Jacobienne

- **Jacobienne :** Pour une fonction $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, la matrice Jacobienne est définie comme :

$$\mathbf{J_f}(\mathbf{x}) = \left[\frac{\partial \mathbf{f}}{\partial x_1}, \frac{\partial \mathbf{f}}{\partial x_2}, \cdots, \frac{\partial \mathbf{f}}{\partial x_n}\right] = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}.$$

- La Jacobienne représente une **approximation linéaire** de comment $\mathbf{f}$ transforme les points dans l'espace d'entrée.
  - **Transformation linéaire locale :** $\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\mathbf{x}_0) + \mathbf{J}(\mathbf{x} - \mathbf{x}_0)$.
- **Déterminant de la Jacobienne :**
  - Mesure l'effet local sur les surfaces ou les volumes.
  - **Interprétation :**
    - $\det(\mathbf{J}) > 0$: La transformation conserve l'orientation.
    - $\det(\mathbf{J}) < 0$: La transformation inverse l'orientation.
    - $|\det(\mathbf{J})|$: Indique l'expansion ou la contraction locale.

La Matrice Jacobienne : Exemple Numérique

- **Fonction considérée :**

$$f(x_1, x_2) = \begin{bmatrix} x_1^2 - x_2 \\ x_1 + x_2^2 \end{bmatrix}.$$

- **Expression de la Jacobienne :**

$$\mathbf{J}_f(x_1, x_2) = \begin{bmatrix} 2x_1 & -1 \\ 1 & 2x_2 \end{bmatrix}.$$

- **Calcul du déterminant de la Jacobienne :**

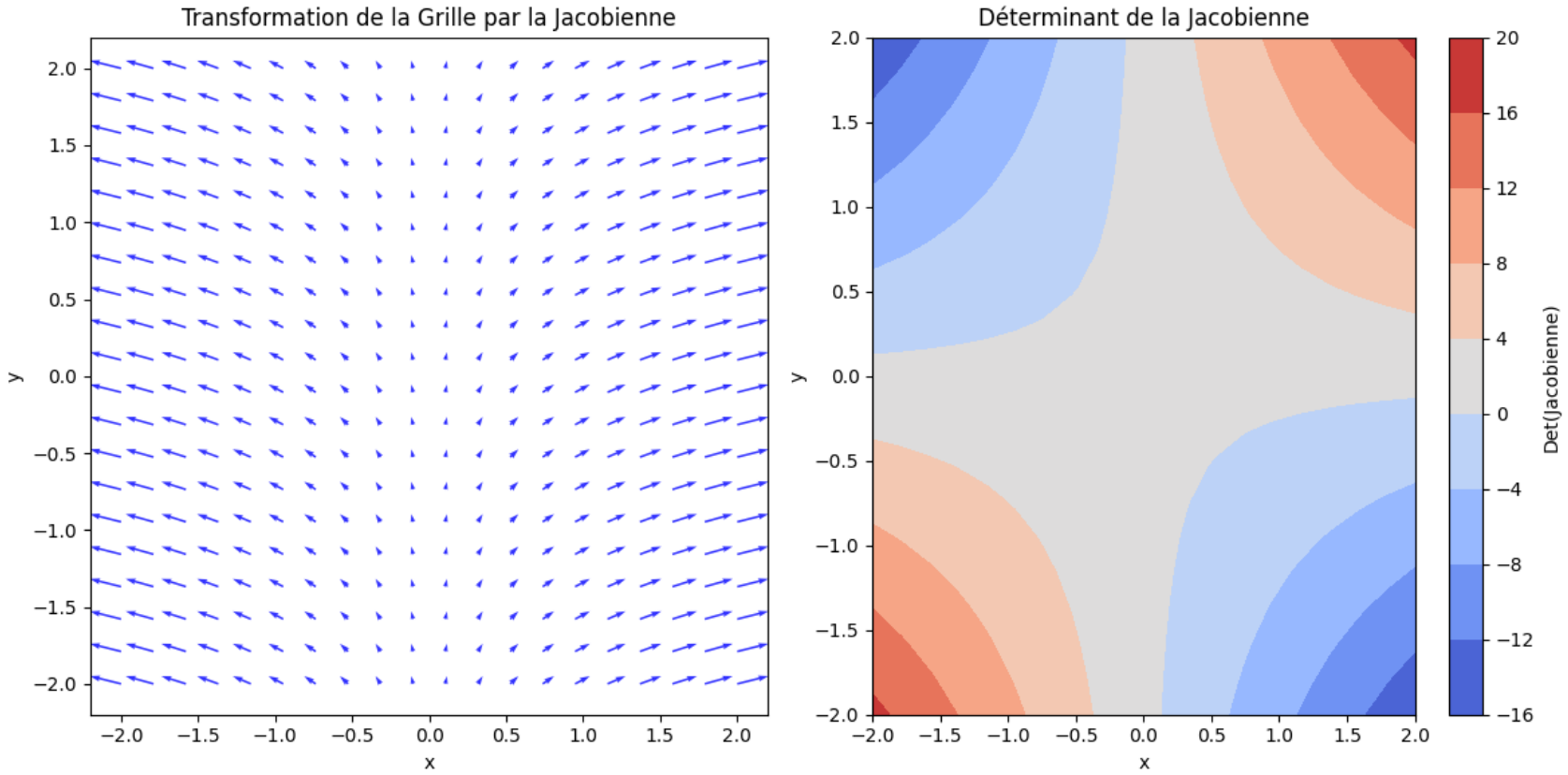
$$\det(\mathbf{J}) = \det \begin{vmatrix} 2x_1 & -1 \\ 1 & 2x_2 \end{vmatrix} = 4x_1x_2 + 1.$$

- En $(x_1, x_2) = (1, 1)$:

$$\mathbf{J}_f(1, 1) = \begin{vmatrix} 2 & -1 \\ 1 & 2 \end{vmatrix}, \quad \det(\mathbf{J}) = 5.$$

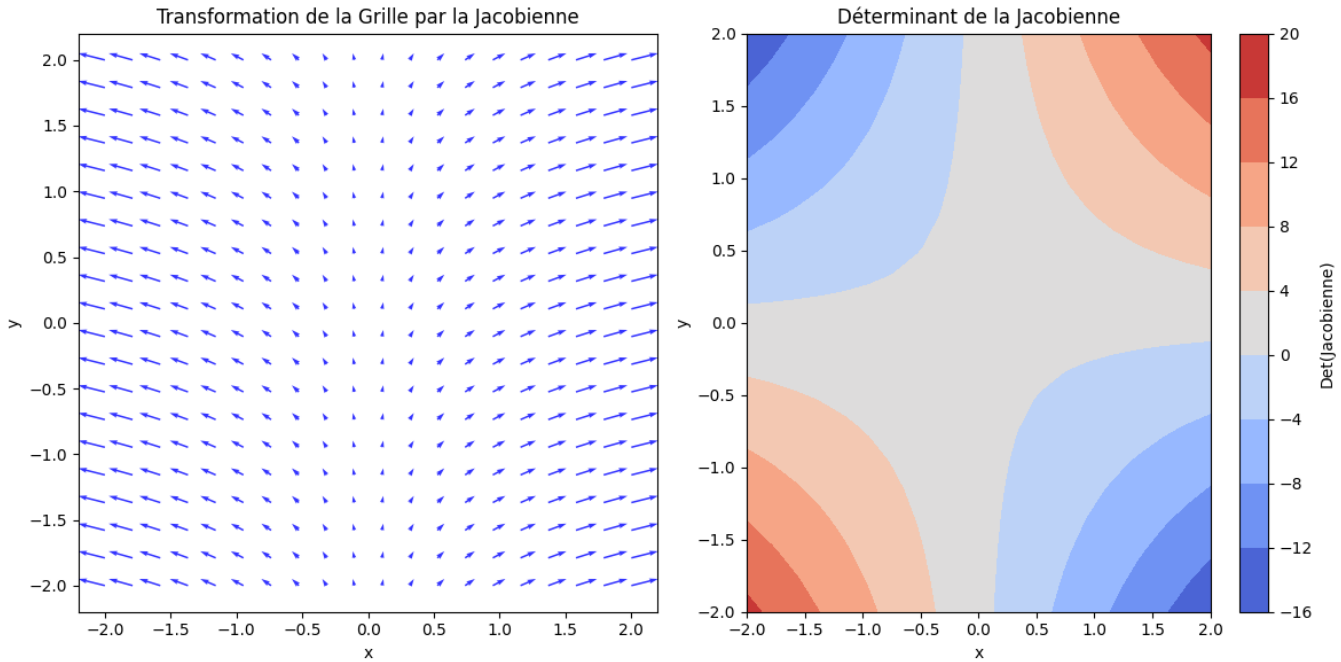
La Matrice Jacobienne : Interprétation Géométrique

- La figure ci-dessous illustre :
  - À gauche : La transformation d'une grille régulière par la Jacobienne.
  - À droite : Le déterminant de la Jacobienne en chaque point.



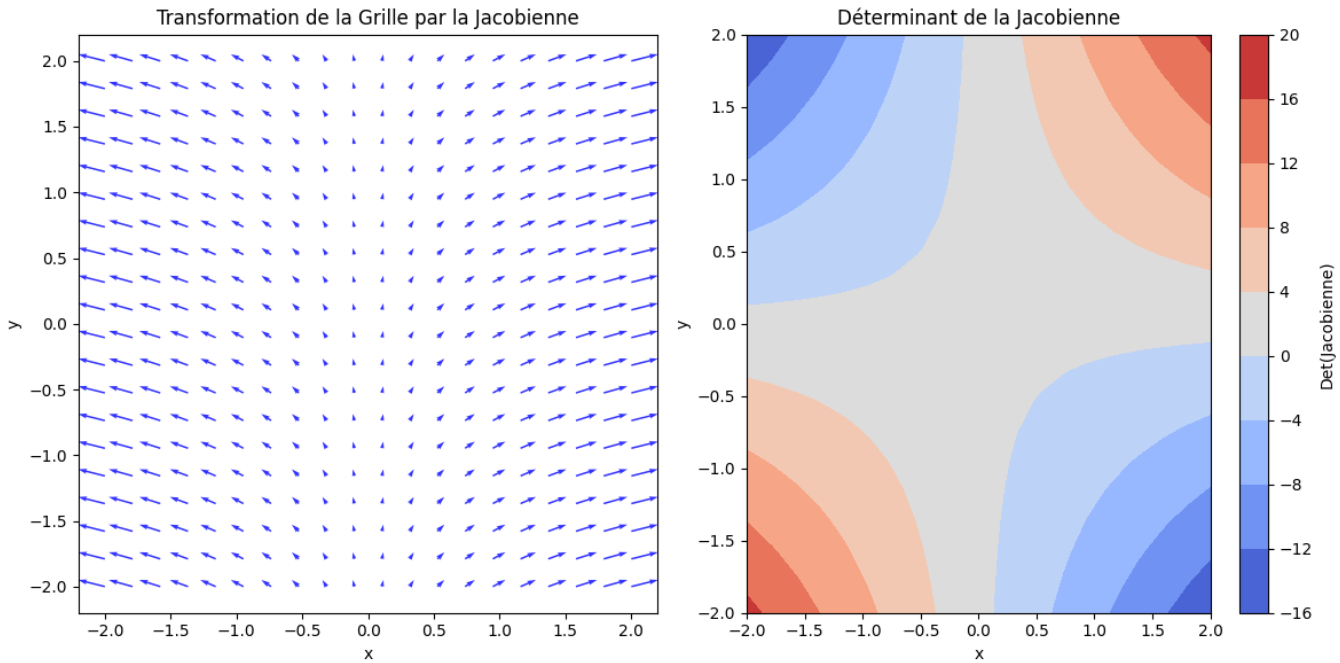
La Matrice Jacobienne : Interprétation Géométrique

- La figure ci-dessous illustre :
  - Le champ de gradient dans un graphe des contours.
  - La direction orthogonale des vecteurs de gradient par rapport aux contours de $f(x_1, x_2)$.
  - Comment la Jacobienne capture la mise à l'échelle et les directions locales.



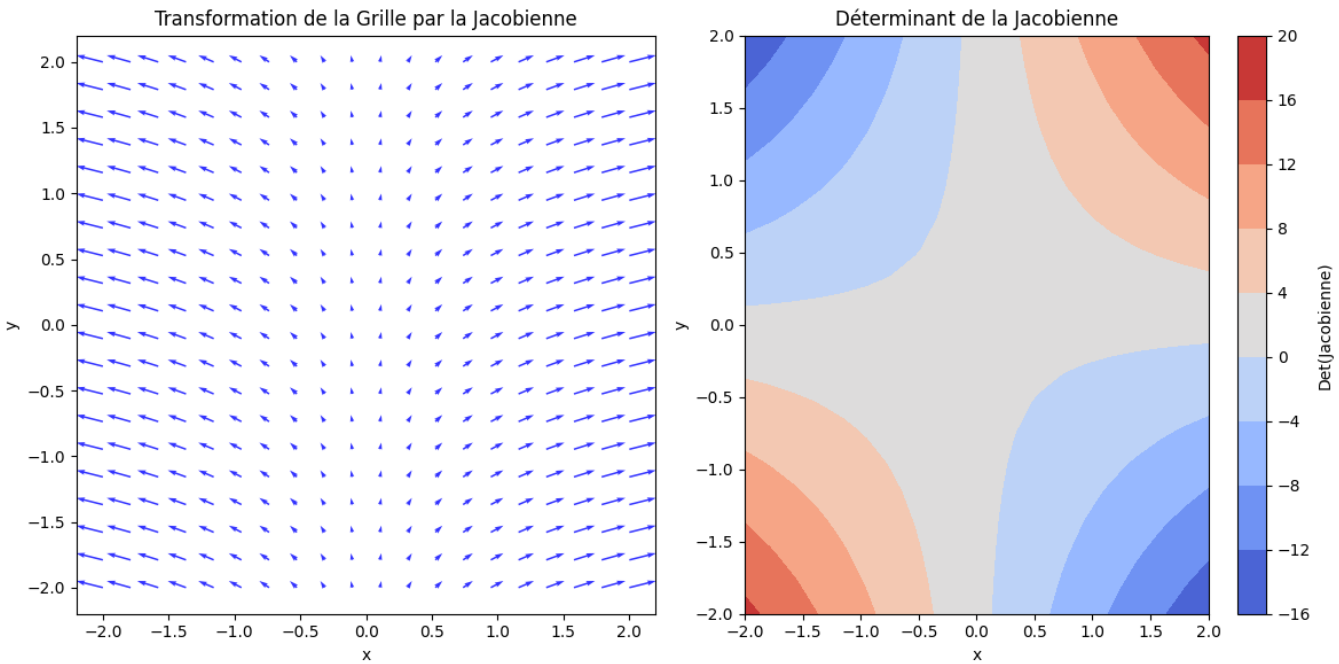
La Matrice Jacobienne : Interprétation Géométrique

- **Transformation locale de l'espace :**
  - La matrice Jacobienne représente une **approximation linéaire** de la manière dont la fonction $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ transforme les points dans l'espace d'entrée.
  - Par exemple, dans $\mathbb{R}^2$, la Jacobienne détermine comment les petites régions autour d'un point sont étirées, comprimées ou tournées.



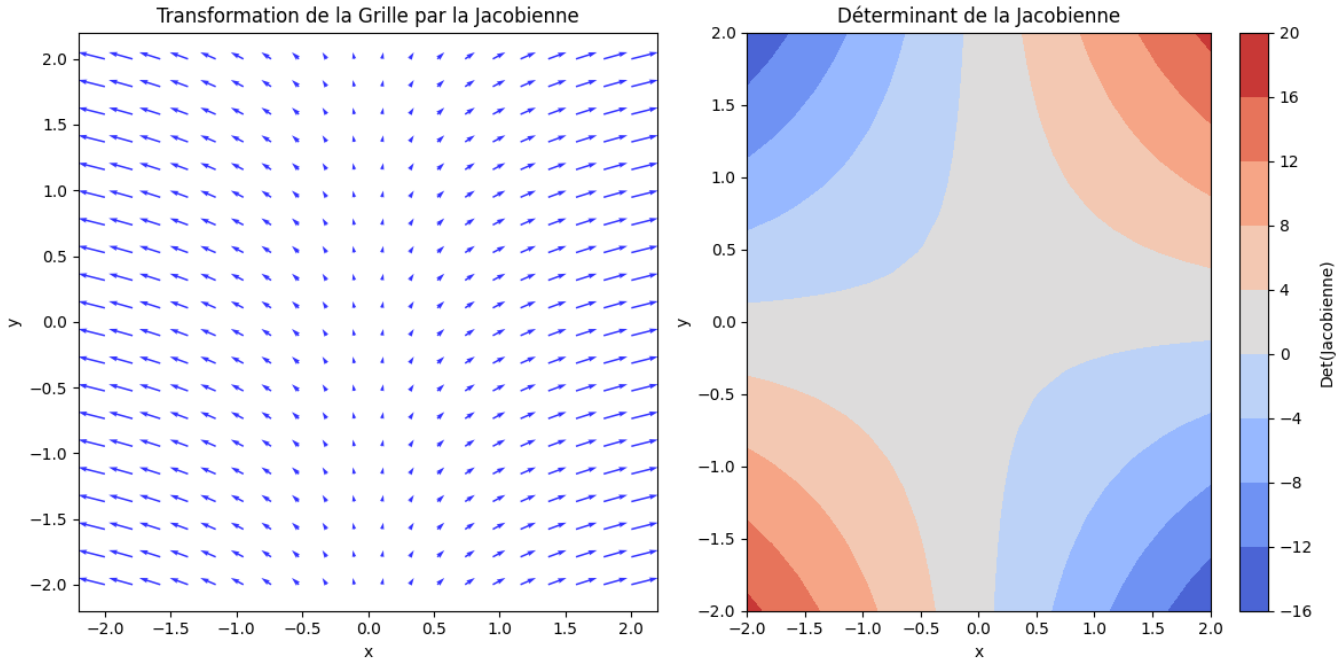
La Matrice Jacobienne : Interprétation Géométrique

- **Directions et mise à l'échelle :**
  - Les lignes de la Jacobienne décrivent le **gradient de chaque composante de sortie** $f_i(x_1, x_2)$.
  - Ces gradients indiquent la direction de la plus grande augmentation pour chaque composante de la fonction.



La Matrice Jacobienne : Interprétation Géométrique

- **Lien avec le champ de gradient :**
  - Pour les fonctions scalaires $f(x_1, x_2)$, la Jacobienne se réduit au gradient $\nabla f(x_1, x_2)$, qui décrit comment le champ scalaire évolue localement.



La Matrice Jacobienne : Cas d'Utilisation

La Jacobienne est essentielle car elle :

- **Adapte les éléments de volume ou de surface** lors de transformations de variables.
- **Assure la conservation des probabilités** dans les changements de variables pour les distributions.
- **Quantifie les effets locaux** d'une transformation sur les volumes ou densités.

En intégration multivariable et en probabilités, la Jacobienne et son déterminant sont donc indispensables pour garantir que les résultats soient corrects dans le nouvel espace transformé.

La Matrice Jacobienne : Code Python

```python
from sympy import symbols, Matrix

# 1. Define symbolic variables
x, y = symbols('x y')

# 2. Define the vector-valued function
f_vector = Matrix([x**2 - y, x + y**2])

# 3. Compute the Jacobian matrix symbolically
jacobian_matrix = f_vector.jacobian([x, y])

# Display the Jacobian matrix
print(f"Jacobian matrix:\n{jacobian_matrix}")

# 4. Evaluate the Jacobian at (1, 1)
point = {x: 1, y: 1}
evaluated_jacobian = jacobian_matrix.subs(point)
print(f"Jacobian matrix at {point}:\n{evaluated_jacobian}")
```

Résultat :

```
Jacobian matrix:
[[2*x, -1]
 [1, 2*y]]
Jacobian matrix at {x: 1, y: 1}:
[[2, -1]
 [1, 2]]
```

La Matrice Hessienne

La Matrice Hessienne

- **Hessienne :** Pour une fonction scalaire $f : \mathbb{R}^n \to \mathbb{R}$, la matrice Hessienne est définie comme la matrice des dérivées secondes :

$$\mathbf{H}_f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

- La matrice Hessienne représente la **courbure locale** de la fonction f.
  - Si $\mathbf{H}$ est définie positive, f a un **minimum local**.
  - Si $\mathbf{H}$ est définie négative, f a un **maximum local**.
  - Si $\mathbf{H}$ a des valeurs propres positives et négatives, f a un **point selle**.
- **Approximation quadratique :** La Hessienne permet d'approximer $f(\mathbf{x})$ localement autour d'un point $\mathbf{x}_0$:

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^\top \mathbf{H}_f (\mathbf{x} - \mathbf{x}_0).$$

Matrice Hessienne : Propriétés

- **Symétrie :** La matrice Hessienne est **symétrique** pour les fonctions $f : \mathbb{R}^n \to \mathbb{R}$ lorsque les dérivées partielles secondes mixtes sont continues :
$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}.$$
- **Semi-définie positive pour les fonctions convexes :** Si f est une fonction **convexe**, alors la matrice Hessienne $\mathbf{H}_f$ est **semi-définie positive** :
$$\mathbf{v}^T \mathbf{H}_f \mathbf{v} \geq 0, \quad \forall\, \mathbf{v} \in \mathbb{R}^n.$$
Si $f(x)$ est **strictement convexe**, alors $\mathbf{H}_f$ est **définie positive** :
$$\mathbf{v}^T \mathbf{H}_f \mathbf{v} > 0, \quad \forall\, \mathbf{v} \neq 0.$$

Matrice Hessienne : Propriétés

- **Semi-définie négative pour les fonctions concaves :** Si f est une fonction **concave**, alors $\mathbf{H}_f$ est **semi-définie négative** :

$$\mathbf{v}^T \mathbf{H}_f \mathbf{v} \leq 0, \quad \forall\, \mathbf{v} \in \mathbb{R}^n.$$

  Si $f(x)$ est **strictement concave**, alors $\mathbf{H}_f$ est **définie négative** :

$$\mathbf{v}^T \mathbf{H}_f \mathbf{v} < 0, \quad \forall\, \mathbf{v} \neq 0.$$

- **Valeurs propres et courbure :** Les valeurs propres de la matrice Hessienne décrivent la courbure locale de la fonction :
  - Si les valeurs propres sont **positives**, f possède un **minimum local**.
  - Si les valeurs propres sont **négatives**, f possède un **maximum local**.
  - Si les valeurs propres sont **mixtes**, f possède un **point selle**.

Matrice Hessienne : Propriétés

- **Forme quadratique :** Pour des petites perturbations $\Delta\mathbf{x}$ proches d'un point $\mathbf{x}$, l'expansion de Taylor au second ordre inclut la Hessienne :

$$f(\mathbf{x}+\Delta\mathbf{x}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^T \Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^T \mathbf{H}_f \Delta\mathbf{x}.$$

  Le terme $\frac{1}{2}\Delta x^T H_f(x)\Delta x$ décrit comment $f(x)$ se courbe localement autour de x.
- **Analyse des points critiques :** Au niveau d'un point critique ($\nabla f(x) = 0$) :
  - Minimum : $\mathbf{H}_f$ est définie positive.
  - Maximum : $\mathbf{H}_f$ est définie négative.
  - Point selle : $\mathbf{H}_f$ a des valeurs propres de signes opposés.

Matrice Hessienne : Propriétés

- **Invariance par l'échelle des vecteurs propres :** Les vecteurs propres de la Hessienne indiquent les directions principales de la courbure, indépendamment de l'échelle.
- **Semi-définie positive dans les moindres carrés :** Dans les problèmes de régression des moindres carrés, la Hessienne de la fonction objectif est toujours **symétrique et semi-définie positive**.
  - $\Rightarrow$ Il suffit donc de trouver le point critique où $\nabla f(\mathbf{x}) = 0$) pour trouver la solution minimale (Minimum).

La Matrice Hessienne : Exemple Numérique

- **Fonction considérée :**

$$f(x_1, x_2) = x_1^2 + 3x_1x_2 + 2x_2^2.$$

- **Expression de la Hessienne :**

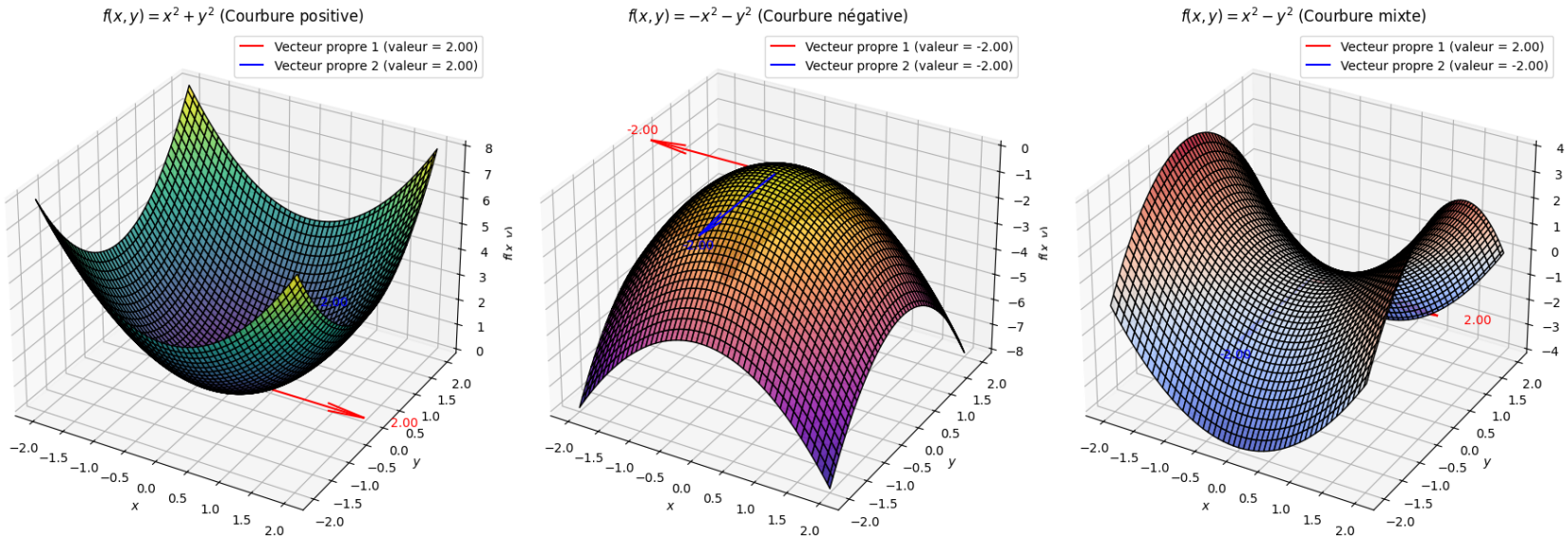
$$\mathbf{H}_f(x_1, x_2) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 3 & 4 \end{bmatrix}.$$

- En $(x_1, x_2) = (1, 2)$, la matrice Hessienne reste constante :

$$\mathbf{H}_f(1, 2) = \begin{bmatrix} 2 & 3 \\ 3 & 4 \end{bmatrix}.$$

La Matrice Hessienne : Interprétation Géométrique

- La matrice Hessienne capture la courbure locale de $f(\mathbf{x})$.
- **Points critiques et Hessienne :**
  - Minimum local : Courbure positive ($\mathbf{H} \geq 0$ (SDP)).
  - Maximum local : Courbure négative ($\mathbf{H} \leq 0$ (SDN)).
  - Point selle : Courbure mixte ($\mathbf{H}$ (indéfinie)).



La Matrice Hessienne : Code Python

```python
from sympy import symbols, hessian

# 1. Define symbolic variables
x1, x2 = symbols('x1 x2')

# 2. Define the scalar function
f = x1**2 + 3*x1*x2 + 2*x2**2

# 3. Compute the Hessian matrix symbolically
hessian_matrix = hessian(f, (x1, x2))

# Display the Hessian matrix
print(f"Hessian matrix:\n{hessian_matrix}")

# 4. Evaluate the Hessian at (1, 2)
point = {x1: 1, x2: 2}
evaluated_hessian = hessian_matrix.subs(point)
print(f"Hessian matrix at {point}:\n{evaluated_hessian}")
```

Résultat :

```
Hessian matrix:
[[2, 3]
 [3, 4]]
Hessian matrix at {x1: 1, x2: 2}:
[[2, 3]
 [3, 4]]
```

Dérivées Matricielles

Dérivées Matricielles : Notation

- Dans le calcul matriciel, deux notations principales sont utilisées pour représenter les dérivées matricielles :
  - la notation de numérateur.
  - la notation de dénominateur.
- **Notation de numérateur :** La dérivée d'un scalaire y par rapport à un vecteur $\mathbf{x} = [x_1, x_2, \ldots, x_n]^{\mathrm{T}}$ est écrite (dans la notation de numérateur) comme suit :

$$\frac{\partial y}{\partial \mathbf{x}} = \left[\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}, \cdots, \frac{\partial y}{\partial x_n}\right].$$

- La dérivée d'un scalaire par rapport à un vecteur $\mathbf{x}$ est donc la transposée du gradient :

$$\nabla f = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}\right] = \left(\frac{\partial f}{\partial \mathbf{x}}\right)^{\mathrm{T}}.$$

Dérivées Matricielles : Notation

- **Notation de dénominateur :** La dérivée d'un scalaire y par rapport à un vecteur $\mathbf{x}$ est écrite (dans la notation de dénominateur) comme suit :

$$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_n} \end{bmatrix} = \left(\frac{\partial f}{\partial \mathbf{x}} \right).$$

- Dans cette notation, la dérivée d'un scalaire par rapport à un vecteur est simplement le gradient sans transposition :

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}.$$

Dérivées Matricielles : Types

Nous allons utliser la notation de numérateur pour les dérivées matricielles. Les six types de dérivées impliquant des scalaires, des vecteurs et des matrices peuvent être organisés comme suit. Ces dérivées apparaissent lorsque l'on considère les relations entre ces objets mathématiques et leurs composantes.

Table: Types de Dérivées Matricielles

Types	**Scalaire** (y)	**Vecteur** $(\mathbf{y})$	**Matrice** $(\mathbf{y})$
Scalaire (x)	$\frac{\partial y}{\partial x}$	$\frac{\partial \mathbf{y}}{\partial x}$	$\frac{\partial \mathbf{Y}}{\partial x}$
Vecteur $(\mathbf{x})$	$\frac{\partial y}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$	–
Matrice $(\mathbf{X})$	$\frac{\partial y}{\partial \mathbf{X}}$	–	–

Dérivées Matricielles : Types

- **Scalaire-à-Scalaire ($\frac{\partial y}{\partial x}$) :** La dérivée classique d'une fonction scalaire par rapport à une variable scalaire.
  - y est un scalaire de taille 1×1,
  - x est un scalaire de taille 1×1.
  - **Dimension de $\frac{\partial y}{\partial x}$:** Scalaire (1×1).
- **Scalaire-à-Vecteur ($\frac{\partial y}{\partial \mathbf{x}}$) :**
  - y est un scalaire de taille 1×1,
  - $\mathbf{x}$ est un vecteur de taille $n \times 1$.
  - **Dimension de $\frac{\partial y}{\partial \mathbf{x}}$:** Vecteur ligne (Transposée du gradient) $(1 \times n)$.
- **Scalaire-à-Matrice ($\frac{\partial y}{\partial \mathbf{X}}$) :** Gradient matriciel, souvent utilisé dans les problèmes d'optimisation avec des variables matricielles.
  - y est un scalaire de taille 1×1,
  - $\mathbf{X}$ est une matrice de taille $m \times n$.
  - **Dimension de $\frac{\partial y}{\partial \mathbf{X}}$:** lignes et colonnes interchangées$(n \times m)$.

Dérivées Matricielles : Types

- **Vecteur-à-Scalaire ($\frac{\partial \mathbf{y}}{\partial x}$) :** Une dérivée décrivant comment chaque composante d'un vecteur dépend d'un scalaire.
  - $\mathbf{y}$ est un vecteur de taille $m \times 1$.
  - **Dimension de $\frac{\partial \mathbf{y}}{\partial x}$:** Vecteur $(m \times 1)$.
- **Vecteur-à-Vecteur ($\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$) :** La matrice Jacobienne, où chaque élément représente une dérivée partielle d'une composante du vecteur par rapport à une variable vectorielle.
  - $\mathbf{y}$ est un vecteur de taille $m \times 1$,
  - $\mathbf{x}$ est un vecteur de taille $n \times 1$.
  - **Dimension de $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$:** Matrice $(m \times n)$.
- **Matrice-à-Scalaire ($\frac{\partial \mathbf{Y}}{\partial x}$) :** Une dérivée de la même dimension que la matrice $\mathbf{Y}$.
  - $\mathbf{Y}$ est une matrice de taille $m \times n$.
  - **Dimension de $\frac{\partial \mathbf{Y}}{\partial x}$:** Matrice $(m \times n)$.

Dérivées Matricielles : Types

- **Dérivée d'une matrice par rapport à un vecteur :**

$$\frac{\partial \mathbf{M}}{\partial \mathbf{v}} \quad \Rightarrow \quad \text{Dimension : } (m \cdot n) \times p.$$

- **Dérivée d'un vecteur par rapport à une matrice :**

$$\frac{\partial \mathbf{v}}{\partial \mathbf{M}} \quad \Rightarrow \quad \text{Dimension : } p \times (m \cdot n).$$

- **Dérivée d'une matrice par rapport à une matrice :**

$$\frac{\partial \mathbf{M}_1}{\partial \mathbf{M}_2} \quad \Rightarrow \quad \text{Dimension : } (m_1 \cdot n_1) \times (m_2 \cdot n_2).$$

- Ces résultats correspondent à des **tenseurs de différents rangs et on en n'aura besoin dans notre cours**.

Dérivées Matricielles : Calcul

- La dérivée d'un vecteur $\mathbf{a}$ **par rapport à un scalaire** x est un vecteur dont les composantes sont données par :

$$\left(\frac{\partial \mathbf{a}}{\partial x}\right)_i = \left(\frac{\partial a_i}{\partial x}\right).$$

- Une définition analogue existe pour la dérivée d'un scalaire ou d'un vecteur **par rapport à un autre vecteur** où les composantes sont données par :

$$\left(\frac{\partial x}{\partial \mathbf{a}}\right)_i = \frac{\partial x}{\partial a_i}, \quad \left(\frac{\partial \mathbf{a}}{\partial \mathbf{b}}\right)_{ij} = \frac{\partial a_i}{\partial b_j}.$$

- **Comment obtenir la formule d'une dérivée matricielle** ?

$$\text{Exemple : } \frac{\partial}{\partial \mathbf{x}}\left(\mathbf{x}^{\mathrm{T}}\mathbf{a}\right).$$

Dérivées Matricielles : Calcul

- **Calculons :** $\frac{\partial}{\partial \mathbf{x}} \left(\mathbf{x}^{\mathrm{T}} \mathbf{a} \right)$, où nous avons:
  - $\mathbf{x}$ est un vecteur colonne de taille $n \times 1$,
  - $\mathbf{a}$ est un vecteur colonne de taille $n \times 1$.
  - $\Rightarrow$ Dérivée Scalaire-à-Vecteur $\Rightarrow$ Dim $\frac{\partial}{\partial \mathbf{x}} \left(\mathbf{x}^{\mathrm{T}} \mathbf{a} \right)$ est de $1 \times n$
- **Étape 1 : Expansion de l'expression en termes de scalaires :**
  - Pour $\mathbf{x}^{\mathrm{T}} \mathbf{a}$: $\mathbf{x}^{\mathrm{T}} \mathbf{a} = \sum_{i=1}^{n} x_i a_i$,
  - où x_i et a_i sont les composantes respectives de $\mathbf{x}$ et $\mathbf{a}$.
- **Étape 2 : Calcul de la dérivée de l'expression en termes de scalaires.**
- **Étape 3 : Réécrire la dérivée calculée en notation matricielle ou vectorielle** (Si c'est possible).

ÉDérivées Matricielles : Calcul

- La dérivée d'une fonction scalaire $f(\mathbf{x})$ par rapport à un vecteur $\mathbf{x}$ est un vecteur dont la i-ème composante est :

$$\left[\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right]_i = \frac{\partial f(\mathbf{x})}{\partial x_i}.$$

- **Calcul :** $f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{a} = \sum_{i=1}^{n} x_i a_i \quad \Rightarrow \quad \frac{\partial}{\partial x_i}\left(\mathbf{x}^{\mathrm{T}}\mathbf{a}\right) = a_i.$
- Donc, la dérivée est :

$$\frac{\partial}{\partial \mathbf{x}}\left(\mathbf{x}^{\mathrm{T}}\mathbf{a}\right) = \left[a_1, a_2, \cdots, a_n\right] = \mathbf{a}^{\mathrm{T}}.$$

- **De même pour.** $\frac{\partial}{\partial \mathbf{a}}\left(\mathbf{a}^{\mathrm{T}}\mathbf{x}\right) = \mathbf{x}^{\mathrm{T}}$

Dérivées Matricielles : Calcul

- **Calculons :** $\frac{\partial}{\partial \mathbf{x}} (\mathbf{Ax})$, où nous avons :
  - $\mathbf{A}$ est une matrice de taille $m \times n$,
  - $\mathbf{x}$ est un vecteur colonne de taille $n \times 1$.
  - $\Rightarrow$ Dérivée Vecteur-à-Vecteur $\Rightarrow$ Dim de $\frac{\partial}{\partial \mathbf{x}} (\mathbf{Ax})$: $m \times n$
- **Étape 1 : Expansion de l'expression en termes de scalaires :**
  - $\mathbf{Ax}$ produit un vecteur colonne de taille $m \times 1$,
  - L'élément i-ième de $\mathbf{Ax}$ est donné par :

$$(\mathbf{Ax})_i = \sum_{j=1}^{n} A_{ij} x_j,$$

où A_{ij} est l'élément de la i-ième ligne et j-ième colonne de $\mathbf{A}$, et x_j est le j-ième composant de $\mathbf{x}$.

- **Étape 2 : Calcul des dérivées en termes de scalaires.**
- **Étape 3 : Réécrire les dérivées en notation matricielle.**

Dérivées Matricielles : Calcul

- **Expansion scalaire :** pour calculer $\frac{\partial}{\partial \mathbf{x}} (\mathbf{Ax})$

$$(\mathbf{Ax})_i = \sum_{j=1}^{n} A_{ij} x_j.$$

- **Calcul de la dérivée :**

$$\left(\frac{\partial \mathbf{Ax}}{\partial \mathbf{x}} \right)_{ik} = \frac{\partial (\mathbf{Ax})_i}{\partial x_k} = A_{ik}.$$

  - Cette dérivée est une matrice de taille $m \times n$ $\Rightarrow$ c'est la matrice $\mathbf{A}$.
- **Résultat matriciel :** La dérivée complète est donnée par :

$$\frac{\partial (\mathbf{Ax})}{\partial \mathbf{x}} = \mathbf{A}.$$

Dérivées Matricielles : Calcul

- **Calculons :** $\frac{\partial}{\partial \mathbf{X}} \left(\mathbf{a}^\top \mathbf{X} \mathbf{b} \right)$, où nous avons :
  - $\mathbf{X}$ est une matrice de taille $n \times n$,
  - $\mathbf{a}$ et $\mathbf{b}$ sont des vecteurs colonne de taille $n \times 1$. $\Rightarrow$ Dérivée de Scalaire-à-Matrice $\Rightarrow$ Dim est de $n \times n$
- **Étape 1 : Expansion de l'expression en termes de scalaires :**
  - $\mathbf{a}^\top \mathbf{X} \mathbf{b} = \sum_{i=1}^{n} \sum_{j=1}^{n} a_i X_{ij} b_j$,
  - où a_i, b_j, et X_{ij} sont les composantes respectives de $\mathbf{a}$, $\mathbf{b}$, et $\mathbf{X}$.
- **Étape 2 : Calcul de la dérivée de l'expression en termes de scalaires.**
- **Étape 3 : Réécrire la dérivée calculée en notation matricielle (si possible).**

Dérivées Matricielles : Calcul

- La dérivée d'une fonction scalaire $f(\mathbf{X})$ par rapport à une matrice $\mathbf{X}$ donne une matrice, dont l'élément (i, j) est :

$$\left[\frac{\partial f(\mathbf{X})}{\partial \mathbf{X}}\right]_{ij} = \frac{\partial f(\mathbf{X})}{\partial X_{ij}}.$$

- **Calcul :**

$$f(\mathbf{X}) = \mathbf{a}^\top \mathbf{X}\mathbf{b} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_i X_{ij} b_j.$$

- En dérivant par rapport à X_{ij}, seul le terme contenant X_{ij} reste :

$$\frac{\partial f(\mathbf{X})}{\partial X_{ij}} = a_i b_j.$$

- La matrice dont l'élément (i, j) est donné par $a_i b_j$ est exprimée par le produit extérieur (outer product) des vecteurs $\mathbf{a}$ et $\mathbf{b}$. Par conséquent, la dérivée est : $\frac{\partial}{\partial \mathbf{X}}\left(\mathbf{a}^\top \mathbf{X}\mathbf{b}\right) = \mathbf{a}\mathbf{b}^{\mathrm{T}}$.

Calcul Matricielle : Formules importantes

- **Dérivée d'un scalaire par rapport à une matrice**
  Si $y = \mathbf{a}^T \mathbf{X} \mathbf{b}$ où $\mathbf{a}$ et $\mathbf{b}$ sont des vecteurs constants et $\mathbf{X}$ est une matrice, alors :
  $$\frac{\partial y}{\partial \mathbf{X}} = \mathbf{a}\mathbf{b}^T$$
- **Dérivée d'une matrice par rapport à elle-même**
  Si $\mathbf{X}$ est une matrice :
  $$\frac{\partial \mathbf{X}}{\partial \mathbf{X}} = \mathbf{I}$$
  où $\mathbf{I}$ est la matrice identité de taille appropriée.
- **Trace d'un produit matriciel**
  Si $\mathbf{A}$ est une matrice constante et $\mathbf{X}$ une matrice variable :
  $$\frac{\partial \text{tr}(\mathbf{A}\mathbf{X})}{\partial \mathbf{X}} = \mathbf{A}^T$$

Calcul Matricielle : Formules importantes

- **Forme quadratique**
  Pour $y = \mathbf{x}^T \mathbf{A}\mathbf{x}$ où $\mathbf{x}$ est un vecteur et $\mathbf{A}$ une matrice symétrique constante :
  $$\frac{\partial y}{\partial \mathbf{x}} = 2\mathbf{A}\mathbf{x}$$
- **Produit matrice-matrice**
  Pour $\mathbf{C} = \mathbf{A}\mathbf{X}\mathbf{B}$ où $\mathbf{A}$ et $\mathbf{B}$ sont des matrices constantes et $\mathbf{X}$ une matrice variable :
  $$\frac{\partial \text{tr}(\mathbf{C})}{\partial \mathbf{X}} = \mathbf{A}^T \mathbf{B}^T$$
- **Inverse d'une matrice**
  Si $\mathbf{X}$ est une matrice inversible :
  $$\frac{\partial \text{tr}(\mathbf{A}\mathbf{X}^{-1})}{\partial \mathbf{X}} = -\mathbf{X}^{-T} \mathbf{A}^T \mathbf{X}^{-T}$$

Dérivées Matricielles : Formules

- Pour le produit de matrices $\mathbf{A}$ et $\mathbf{B}$, nous avons :

$$\frac{\partial}{\partial x}(\mathbf{AB}) = \frac{\partial \mathbf{A}}{\partial x}\mathbf{B} + \mathbf{A}\frac{\partial \mathbf{B}}{\partial x}.$$

- La dérivée de l'inverse d'une matrice $\mathbf{A}$ peut être exprimée comme suit :

$$\frac{\partial}{\partial x}(\mathbf{A}^{-1}) = -\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\mathbf{A}^{-1}.$$

- La dérivée du logarithme du déterminant d'une matrice $\mathbf{A}$ est donnée par :

$$\frac{\partial}{\partial x}\ln|\mathbf{A}| = \mathrm{Tr}\left(\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\right).$$

- Si $\mathbf{A}$ et $\mathbf{B}$ sont des matrices, alors :

$$\frac{\partial}{\partial \mathbf{A}}\mathrm{Tr}(\mathbf{AB}) = \mathbf{B}^\top.$$

Dérivées Matricielles : Formules

- En utilisant cette notation, les propriétés suivantes sont valides :

$$\frac{\partial}{\partial \mathbf{A}} \mathrm{Tr}(\mathbf{A}^\top \mathbf{B}) = \mathbf{B}, \quad \frac{\partial}{\partial \mathbf{A}} \mathrm{Tr}(\mathbf{A}) = \mathbf{I}.$$

- Pour une matrice symétrique $\mathbf{A}$:

$$\frac{\partial}{\partial \mathbf{A}} \mathrm{Tr}(\mathbf{A}\mathbf{B}\mathbf{A}^\top) = \mathbf{A}(\mathbf{B} + \mathbf{B}^\top).$$

- La dérivée du logarithme du déterminant est donnée par :

$$\frac{\partial}{\partial \mathbf{A}} \ln |\mathbf{A}| = (\mathbf{A}^{-1})^\top.$$

- **Norme de Frobenius d'une matrice**
  Pour la norme de Frobenius au carré, $\|\mathbf{X}\|_F^2 = \mathrm{tr}(\mathbf{X}^T\mathbf{X})$, la dérivée par rapport à $\mathbf{X}$ est :

$$\frac{\partial \|\mathbf{X}\|_F^2}{\partial \mathbf{X}} = 2\mathbf{X}$$

- Pour une revue plus complète, consultez : *The Matrix Cookbook*.

Table des Matières

Rappel sur l'Optimisation

Formulation du Problème d'Optimisation

Formulation du Problème d'Optimisation

- **Définition :** L'optimisation consiste à trouver les valeurs des variables d'entrée $\mathbf{x}$ qui minimisent (ou maximisent) une fonction objectif $f(\mathbf{x})$.
- **Objectif :**
  - Trouver un **minimum global** ou un **minimum local**.
  - Résoudre :
$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x}).$$
- **Applications :**
  - Régression linéaire, entraînement des réseaux de neurones.
- **Formulations des problèmes d'optimisation :**
  - **Sans contrainte :** $\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x})$, où $\mathbf{x} \in \mathbb{R}^n$ et $f : \mathbb{R}^n \to \mathbb{R}$.
  - **Avec contrainte(s) :** $\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x})$, **sous les contraintes :** $g_i(\mathbf{x}) \leq 0$ **(contraintes d'inégalité)**, $h_j(\mathbf{x}) = 0$ **(contraintes d'égalité)**. où g_i et h_j des fonctions scalaires.

Solutions Analytiques pour l'Optimisation

Optimisation : Solutions Analytiques

- **Concept Fondamental :**
  - Une solution analytique est obtenue en résolvant directement les équations de l'objectif.
  - Par exemple, pour une fonction quadratique $f(x) = ax^2 + bx + c$, la solution est obtenue en résolvant :

$$\frac{df}{dx} = 2ax + b = 0.$$

- **Interprétation Géométrique :**
  - Le point critique (x^*) est le point où la pente de la courbe est nulle $(\frac{df}{dx} = 0)$.
  - Ce point représente un minimum ou un maximum en fonction de la concavité de $f(x)$.
- **Limites :**
  - Difficile pour des fonctions non linéaires, non convexes ou en haute dimension.
  - Sensible aux matrices mal conditionnées.

Solutions Analytiques : Exemple Numérique

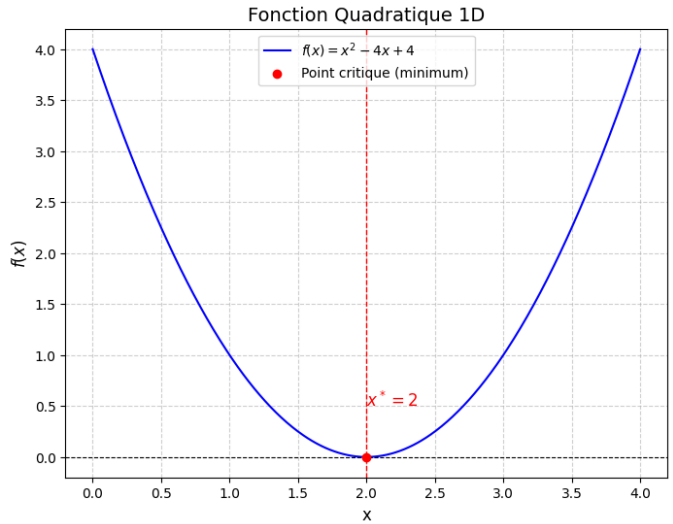
- **Fonction à optimiser :**

$$f(x) = x^2 - 4x + 4$$

- **Solution analytique :**

$$\frac{df}{dx} = 2x - 4 = 0 \implies x^* = 2$$

- **Résultat :** Le minimum se trouve en $x = 2$, et $f(2) = 0$.



Solutions Analytiques : Limites

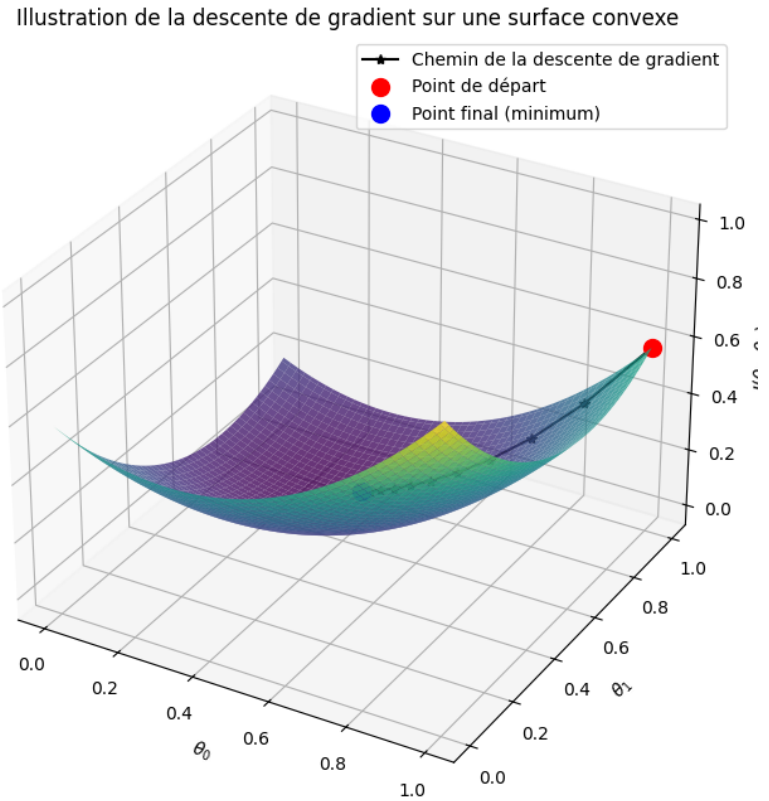
- **Limites des solutions analytiques :**
  - Peuvent être difficiles, voire impossibles, à obtenir pour :
    - Des fonctions non convexes ou non différentiables.
    - Des espaces de recherche de très grande dimension.
    - Des systèmes avec contraintes complexes.
  - Nécessitent des inversions de matrices coûteuses dans certains cas, comme la régression linéaire.
- **Exemples de défis :**
  - **Optimisation non convexe :** Les fonctions ayant des minima locaux rendent les solutions analytiques inutilisables.
  - **Fonctions avec bruit :** Les objectifs bruités ou irréguliers compliquent le calcul exact.
- **Solution alternative :** Utiliser des approches **itératives** comme les méthodes basées sur le gradient.
- **Intuition :**
  - Avancer dans la direction où la fonction diminue le plus rapidement.
  - Approcher le minimum global en utilisant des techniques numériques.

Méthodes d'Optimisation Basées sur le Gradient

Optimisation Basée sur le Gradient

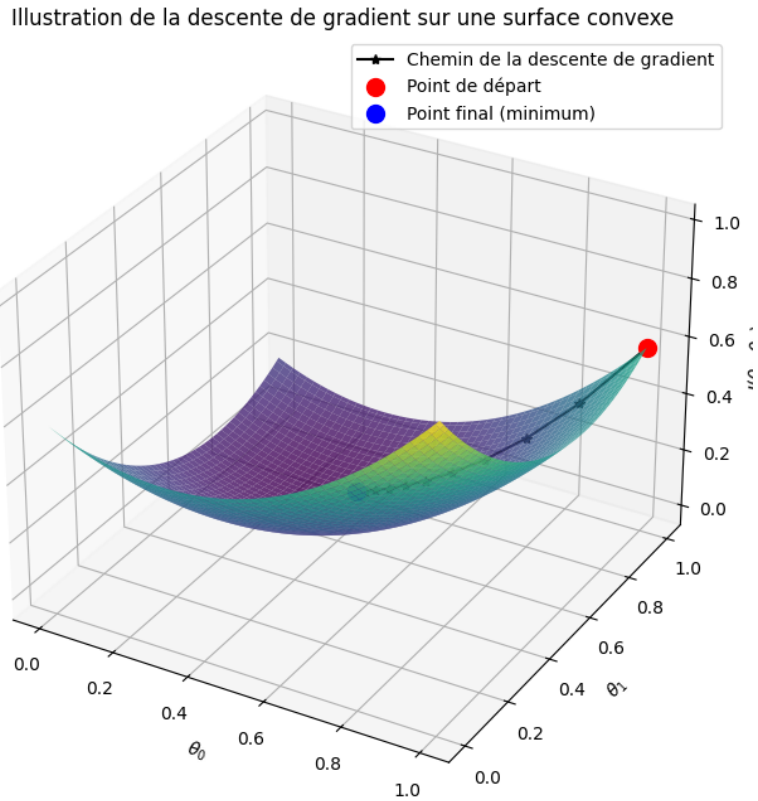
- **Objectif des méthodes basées sur le gradient :**
  - Trouver une solution approximative pour les problèmes difficiles à résoudre analytiquement.
  - Minimiser une fonction $f(\mathbf{x})$ en ajustant $\mathbf{x}$ itérativement.
- **Principe clé :**
  - Utiliser les dérivées (le gradient) pour identifier la direction de la descente la plus rapide.
  - Approcher le minimum de $f(\mathbf{x})$ en suivant cette direction.
- **Pourquoi cela fonctionne :**
  - Le gradient d'une fonction donne la pente locale dans chaque direction.
  - En inversant la direction du gradient, on descend vers un minimum.
- **Applications :**
  - Entraînement des réseaux de neurones.
  - Régressions logistiques et non linéaires.
  - Résolution de problèmes de grande dimension.

Optimisation Basée sur le Gradient



- **Le gradient :**
  - Le gradient est un vecteur qui pointe dans la direction de la plus forte augmentation de la fonction $f(\mathbf{x})$. En suivant la direction opposée $(-\nabla f(\mathbf{x}))$ **on s'adresse vers un minimum.**

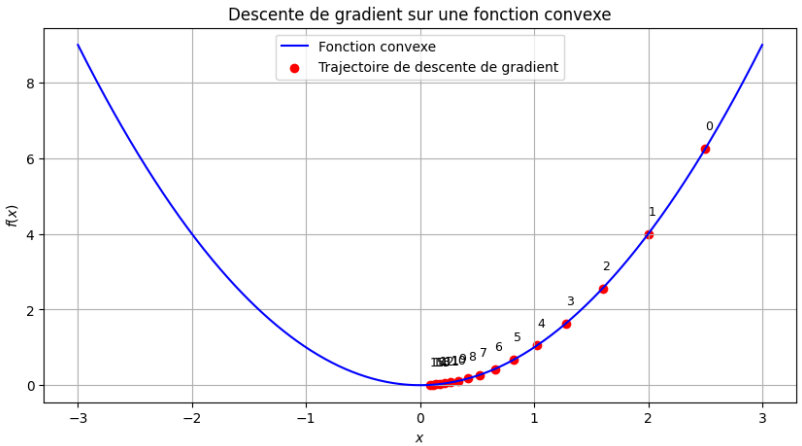
Optimisation Basée sur le Gradient



- **Descente du gradient :**
  - Calcul iteratif : $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla f(\mathbf{x}_t)$, où η est le **taux d'apprentissage**. Il ajuste les valeurs de $\mathbf{x}$ jusqu'à ce qu'un minimum soit atteint.

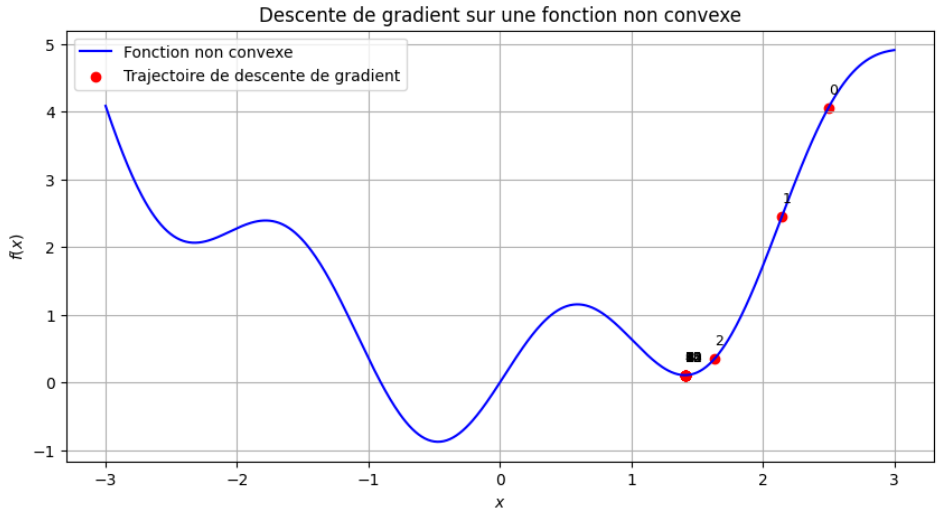
Optimisation Basée sur le Gradient : f Convexe

- **Caractéristiques des fonctions convexes :**
  - La fonction possède un unique minimum global.
  - Le gradient pointe toujours dans la direction menant au minimum global.
- **Illustration :**
  - La descente de gradient converge vers le minimum global sans ambiguïté et les itérations suivent une trajectoire descendante régulière vers le point optimal.



Optimisation Basée sur le Gradient : f Non Convexe

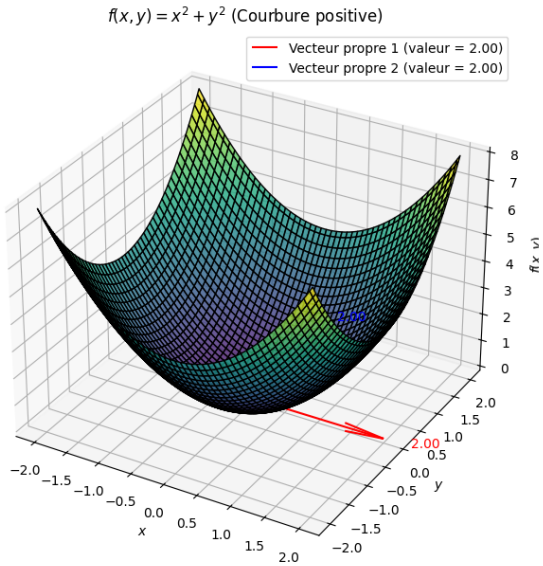
- **Problèmes rencontrés :**
  - **Minima locaux :** Difficulté à échapper aux pièges des minima locaux.
  - **Plateaux :** Zones de faible pente où le gradient est proche de zéro, ralentissant la convergence.
  - **Points de selle :** Direction du gradient n'indique pas clairement une progression vers le minimum.

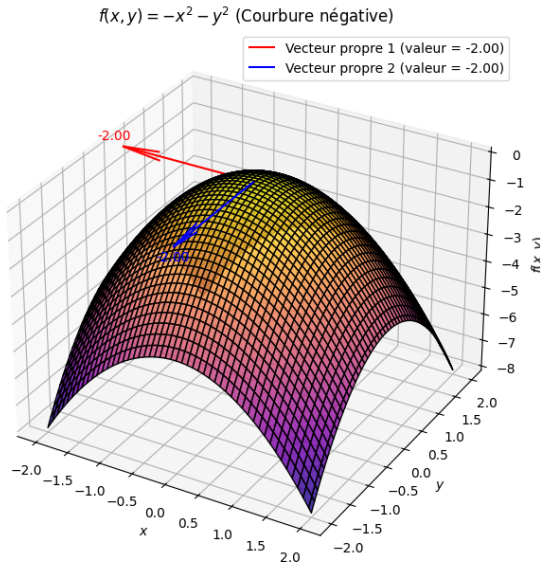


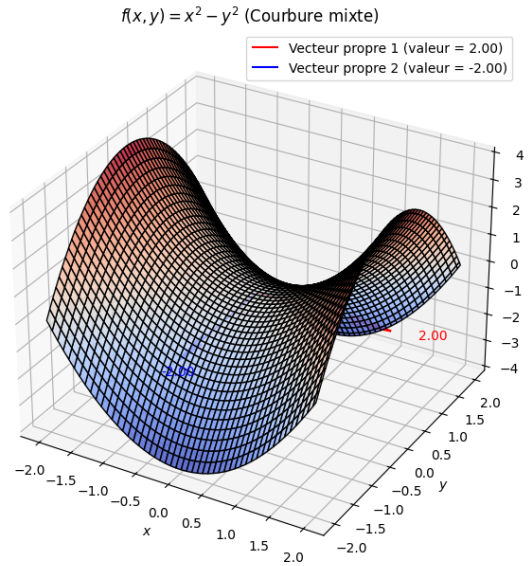
Au-Delà du Gradient : La Hessienne et la Courbure

Au-Delà du Gradient : La Hessienne et la Courbure

- Les gradients indiquent la direction de descente la plus rapide, mais ne capturent pas :
  - La courbure de la fonction.
  - Les zones problématiques comme les plateaux, les minima locaux ou les points de selle.
- La Hessienne apporte des informations supplémentaires :
  - La Hessienne Fournit des informations sur la courbure et aide à identifier les minima locaux, maxima et points de selle.







Au-Delà du Gradient : La Hessienne et la Courbure

- **Définition de la Hessienne :**
  - La Hessienne, notée $\mathbf{H}(\mathbf{x})$, est la matrice des dérivées secondes d'une fonction scalaire $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$.
  - Chaque élément $H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$:

$$\mathbf{H}_f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

- **Interprétation :**
  - La Hessienne donne des informations sur la courbure locale de $f(\mathbf{x})$.
  - Ses valeurs propres (λ_i) déterminent le type de courbure :
    - $\lambda_i > 0$: Courbure positive (minimum local).
    - $\lambda_i < 0$: Courbure négative (maximum local).
    - λ_i mixtes : Points de selle.

Au-Delà du Gradient : La Hessienne et la Courbure

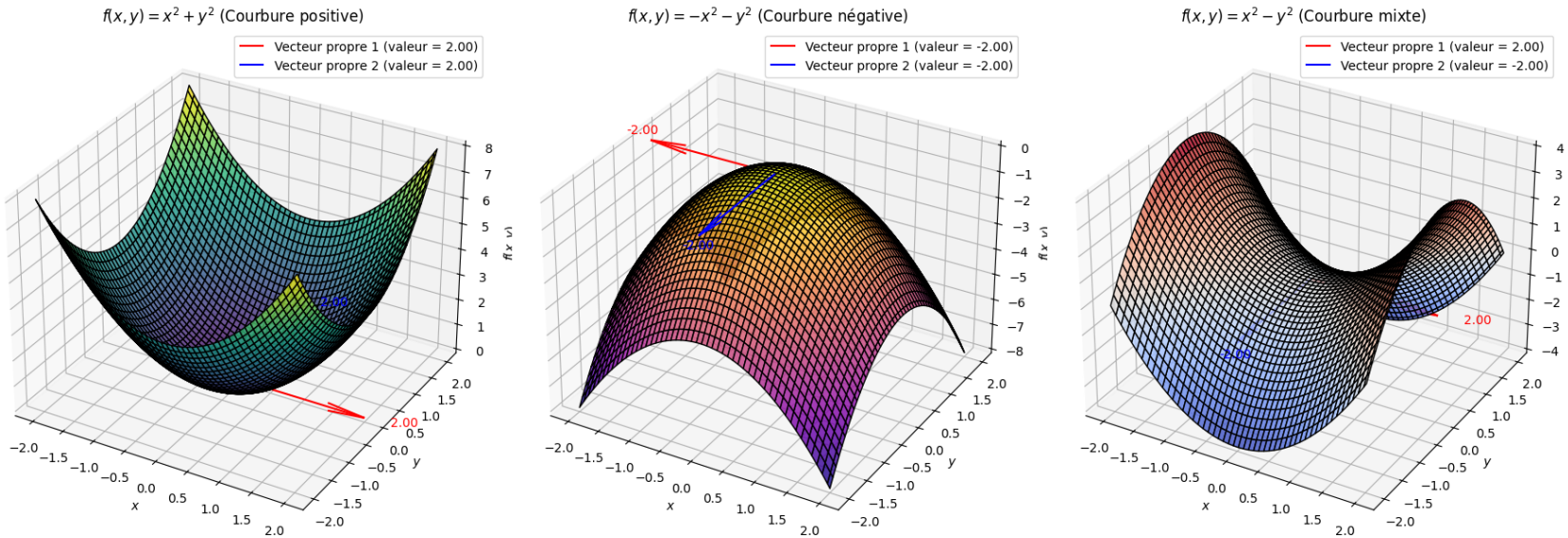
- **Approximation locale :**

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^T(\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T \mathbf{H}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0).$$

- **Interprétation :**
  - $\nabla f(\mathbf{x}_0)$: Indique la direction et la pente locale (gradient).
  - $\mathbf{H}(\mathbf{x}_0)$: Décrit la courbure locale pour ajuster les mises à jour.
- **Lien avec la descente de gradient :**
  - Le gradient seul peut ne pas suffire pour converger rapidement, surtout dans des zones de courbure complexe.
  - La Hessienne ajuste les mises à jour en fonction de la géométrie locale.
- **Avantages des dérivées secondes :**
  - Accélération possible grâce à des tailles de pas ajustées selon la courbure locale.
  - $\eta^* = \frac{\|\nabla f(\mathbf{x})\|^2}{\nabla f(\mathbf{x})^T \mathbf{H} \nabla f(\mathbf{x})}$: Taille de pas optimale en cas de courbure quadratique.

Hessienne : Propriétés en Optimisation

- **Convergence et Courbure :**
  - Les directions principales de la courbure sont données par les vecteurs propres de la Hessienne.
  - Les valeurs propres λ_i indiquent la raideur dans chaque direction.
- **Conditionnement et Convergence :**
  - Le **nombre de conditionnement** de la Hessienne ($\frac{\lambda_{\text{max}}}{\lambda_{\text{min}}}$) indique si la courbure est mal conditionnée.
  - Un mauvais conditionnement ralentit la descente de gradient et mène à des oscillations dans les vallées étroites.



Méthode de Newton

Méthode de Newton

- **Motivation :**
  - Le gradient seul peut ne pas suffire pour converger rapidement, surtout dans des zones de courbure complexe.
  - La Méthode de Newton utilise la courbure locale (**Hessienne**) pour ajuster les pas de manière plus efficace.
- **Formule d'itération :**

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \mathbf{H}(\mathbf{x}_t)^{-1} \nabla f(\mathbf{x}_t),$$

où :

- $\nabla f(\mathbf{x}_t)$ est le gradient.
- $\mathbf{H}(\mathbf{x}_t)$ est la matrice Hessienne (dérivées secondes).

Méthode de Newton

- **Avantages :**
  - Convergence quadratique près d'un minimum.
  - Prend en compte la courbure locale, ce qui améliore les mises à jour dans les vallées étroites.
  - Pas de besoin explicite d'un taux d'apprentissage (η).
- **Inconvénients :**
  - Calcul de $\mathbf{H}^{-1}$ coûteux pour les grandes dimensions.
  - Risque de divergence si la Hessienne n'est pas définie positive.
  - Moins utile pour des fonctions non convexes.

Méthode de Newton

- La descente de gradient suit une trajectoire influencée uniquement par la pente locale.
- La méthode de Newton ajuste ses pas en fonction de la géométrie locale donnée par la Hessienne.

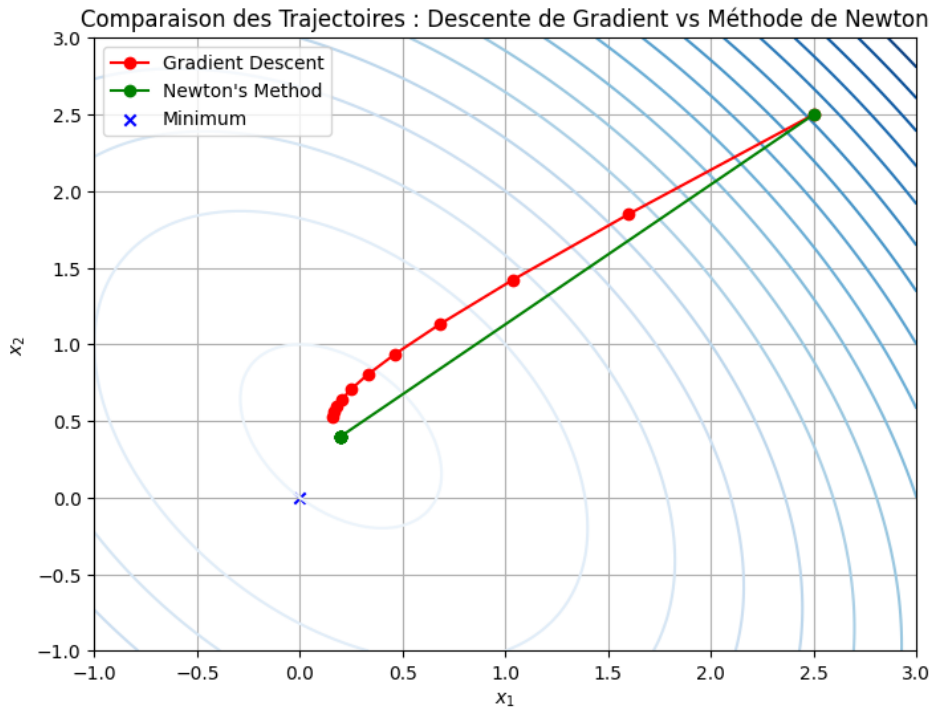


Table des Matières

Table des Matières

Merci!

E-mail: chiheb.trabelsi@polymtl.ca