Calcul matriciel et pivot de Gauss

Informatique pour tous

Questions

On exécute:

```
L1 = [1, 2]
L2 = L1
L2.append(3)
print(L1)
```

Qu'est ce qui est affiché ?

Questions

On exécute:

```
L1 = [1, 2]
L2 = L1
L2.append(3)
print(L1)
```

Qu'est ce qui est affiché ? `[1, 2, 3]`

Questions

On exécute:

```
a = 1
b = a
a = 2
print(b)
```

Qu'est ce qui est affiché ?

Questions

On exécute:

```
a = 1
b = a
a = 2
print(b)
```

Qu'est ce qui est affiché ? `1`

Questions

On exécute:

```
L1 = [1, 2]
L2 = L1[:]
L2.append(3)
print(L1)
```

Qu'est ce qui est affiché ?

Questions

On exécute:

```
L1 = [1, 2]
L2 = L1[:]
L2.append(3)
print(L1)
```

Qu'est ce qui est affiché ? `[1, 2]`

Questions

On exécute:

```
def f(L):
    L.append(3)
L1 = [1, 2]
f(L1)
print(L1)
```

Qu'est ce qui est affiché ?

Questions

On exécute:

```
def f(L):
    L.append(3)
L1 = [1, 2]
f(L1)
print(L1)
```

Qu'est ce qui est affiché ? `[1, 2, 3]`

Questions

On exécute:

```
def f(x):
    x = 2
a = 1
f(a)
print(a)
```

Qu'est ce qui est affiché ?

Questions

On exécute:

```
def f(x):
    x = 2
a = 1
f(a)
print(a)
```

Qu'est ce qui est affiché ? 1

Bilan

Les listes (et les tableaux numpy), lorsque assignés à une autre liste (ou tableau) représentent le **même objet**.

Au contraire, les types de bases (`int`, `float`...) sont **copiés**.

Pour réaliser une copie d'une liste `L` on écrira `L[:]` ou `L.copy()`.

Listes de listes

Si `L = [ [1, 2], ["a", True, 4.2] ]` est une liste de listes:

1. `L[1]` est la liste `["a", True, 4.2]`
2. `L[1][2]` est l'élément d'indice 2 de la liste `L[1]`: `4.2`

Il en est de même pour les tableaux `T` numpy, avec une autre syntaxe possible:

```
T[i, j]
```

Créer une matrice avec des listes

On souhaite créer une matrice 4×4 remplie de `False`:

Créer une matrice avec des listes

On souhaite créer une matrice 4×4 remplie de `False`:

```
M = []
L = [False] * 4
for i in range(4):
    M.append(L)
```

Quel est le problème?

Créer une matrice avec des listes

Code correct:

```
M = []
for i in range(4):
    L = [False] * 4
    M.append(L)
```

Créer une matrice avec des tableaux

Pour créer un tableau numpy à partir d'une liste `L`:

```
np.array(L)
```

Si L est une liste de listes alors `np.array(L)` sera un tableau de tableaux...

Manipulation de matrice

Si `M` est une matrice (liste de listes ou tableau de tableaux):

- L'élément sur la $i$ème ligne, $j$ème colonne est:

Manipulation de matrice

Si `M` est une matrice (liste de listes ou tableau de tableaux):

- L'élément sur la *i*ème ligne, *j*ème colonne est: `M[i][j]`
- Le nombre de lignes de `M` est:

Manipulation de matrice

Si `M` est une matrice (liste de listes ou tableau de tableaux):

- L'élément sur la *i*ème ligne, *j*ème colonne est: `M[i][j]`
- Le nombre de lignes de `M` est: `len(M)`
- Le nombre de colonnes de `M` est:

Manipulation de matrice

Si `M` est une matrice (liste de listes ou tableau de tableaux):

- L'élément sur la *i*ème ligne, *j*ème colonne est: `M[i][j]`
- Le nombre de lignes de `M` est: `len(M)`
- Le nombre de colonnes de `M` est: `len(M[0])`

Extraire des lignes ou des colonnes

On considère la matrice `M` suivante:

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Que vaut `M[1]`?

Extraire des lignes ou des colonnes

On considère la matrice `M` suivante:

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Que vaut `M[1]`?

`array([3, 4, 5])`
Sélectionne la ligne d'indice 1.

Extraire des lignes ou des colonnes

On considère la matrice `M` suivante:

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Que vaut `M[1:3]` ?

Extraire des lignes ou des colonnes

On considère la matrice `M` suivante:

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Que vaut `M[1:3]`?

```
array([[3, 4, 5],
       [6, 7, 8]])
```

Sélectionne les lignes d'indices 1 et 2.

Modifier des lignes ou des colonnes

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Comment remplacer la ligne d'indice 1 par `[0, 0, 0]`?

Modifier des lignes ou des colonnes

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Comment remplacer la ligne d'indice 1 par `[0, 0, 0]`?

```
M[1] = np.array([0, 0, 0])
```

Modifier des lignes ou des colonnes

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Comment multiplier la ligne d'indice 2 par 7?
(opération de dilatation: $L_2 \leftarrow 7 \times L_2$)

Modifier des lignes ou des colonnes

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Comment multiplier la ligne d'indice 2 par 7?
(opération de dilatation: $L_2 \leftarrow 7 \times L_2$)

```
M[2] = 7*M[2]
```

Échanger deux lignes

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Comment échanger les lignes d'indices 0 et 1?

Échanger deux lignes

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Comment échanger les lignes d'indices 0 et 1?

```
tmp = M[0]
M[0] = M[1]
M[1] = tmp
```

Échanger deux lignes

```
tmp = M[0]
M[0] = M[1]
M[1] = tmp
```

`M` est alors égale à:

```
array([[3, 4, 5],
       [3, 4, 5],
       [6, 7, 8]])
```

Échanger deux lignes

```
tmp = M[0]
M[0] = M[1]
M[1] = tmp
```

`M` est alors égale à:

```
array([[3, 4, 5],
       [3, 4, 5],
       [6, 7, 8]])
```

Problème: `M[0] = M[1]` modifie aussi `tmp`!

Échanger deux lignes

Contrairement aux listes, `[:]` ne réalise pas une copie d'un tableau...

On peut utiliser la méthode `copy` à la place: `T.copy()`.

Échanger deux lignes

```
In [55]: tmp = M[0].copy()

In [56]: M[0] = M[1].copy()

In [57]: M[1] = tmp

In [58]: M
Out[58]:
array([[3, 4, 5],
       [0, 1, 2],
       [6, 7, 8]])
```

Échanger deux lignes

On peut aussi utiliser la possibilité, en Python, d'assigner plusieurs variables en parallèle:

```
a, b = c, d
```

`a` et `b` prennent alors simultanément les valeurs de `c` et `d`.

Échanger deux lignes

On peut donc échanger les lignes `i` et `j` de `M` en écrivant:

Échanger deux lignes

On peut donc échanger les lignes `i` et `j` de `M` en écrivant:

```
M[i], M[j] = M[j].copy(), M[i].copy()
```

Méthode du pivot de Gauss

On veut résoudre un système linéaire de n équations à n inconnues de la forme:

$$(S): \begin{cases} a_{1,1}\, x_1 + \cdots + a_{1,n}\, x_n = b_1 \\ \qquad\qquad \cdots \\ a_{n,1}\, x_1 + \cdots + a_{n,n}\, x_n = b_n \end{cases}$$

Méthode du pivot de Gauss

On veut résoudre un système linéaire de n équations à n inconnues de la forme:

$$(\text{S}): \begin{cases} a_{1,1}\, x_1 + \cdots + a_{1,n}\, x_n = b_1 \\ \cdots \\ a_{n,1}\, x_1 + \cdots + a_{n,n}\, x_n = b_n \end{cases}$$

On peut écrire ces équations sous forme matricielle :

$$(S) \Leftrightarrow AX = B$$

$$\text{avec} \quad A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix}, \quad X = \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{pmatrix}$$

Méthode du pivot de Gauss

La méthode du pivot de Gauss comporte 2 grandes étapes :

1. **échelonnement du système** (descente),
2. **réduction du système** (remontée).

Méthode du pivot de Gauss

La méthode du pivot de Gauss comporte 2 grandes étapes :

1. **échelonnement du système** (descente),
2. **réduction du système** (remontée).

Étapes réalisées avec des **opérations élémentaires sur les lignes**:

- $L_i \leftarrow \lambda L_i$ avec $\lambda \neq 0$,
- $L_j \leftarrow L_j + \lambda L_i$ avec $i \neq j$,
- $L_i \leftrightarrow L_j$.

Méthode du pivot de Gauss

La méthode du pivot de Gauss comporte 2 grandes étapes :

1. **échelonnement du système** (descente),
2. **réduction du système** (remontée).

Étapes réalisées avec des **opérations élémentaires sur les lignes**:

- $L_i \leftarrow \lambda L_i$ avec $\lambda \neq 0$,
- $L_j \leftarrow L_j + \lambda L_i$ avec $i \neq j$,
- $L_i \leftrightarrow L_j$.

Appliquer des opérations élémentaires à un système d'équations ne change pas ses solutions.

Matrice augmentée

$$(\mathrm{S}):\ \begin{cases} a_{1,1}\,x_1 + \cdots + a_{1,n}\,x_n = b_1 \\ \qquad\qquad \cdots \\ a_{n,1}\,x_1 + \cdots + a_{n,n}\,x_n = b_n \end{cases}$$

Il est pratique de considérer la **matrice augmentée** du système (S):

$$(A|B) = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} & b_1 \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} & b_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} & b_n \end{pmatrix}$$

Méthode du pivot de Gauss

On veut écrire un algorithme qui:

1. Renvoie l'unique solution de $AX = B$, si A est inversible.

Méthode du pivot de Gauss

On veut écrire un algorithme qui:

1. Renvoie l'unique solution de $AX = B$, si A est inversible.
2. Sinon, indique que A n'est pas inversible (il peut donc exister aucune solution ou une infinité de solutions).

Descente

$$(A|B) = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} & b_1 \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} & b_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} & b_n \end{pmatrix}$$

1. Première étape :
   - Si nécessaire, échanger 2 lignes de façon à avoir $a_{1,1} \neq 0$.

Descente

$$(A|B) = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} & b_1 \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} & b_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} & b_n \end{pmatrix}$$

1. Première étape :
   - Si nécessaire, échanger 2 lignes de façon à avoir $a_{1,1} \neq 0$.
   - On effectue des opérations $L_i \leftarrow L_i + \lambda L_1$ pour i allant de 2 à n de manière à mettre des zéros sur la 1ère colonne, en dessous de la diagonale.

Descente

$$(A|B) = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} & b_1 \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} & b_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} & b_n \end{pmatrix}$$

1. Première étape :
   - Si nécessaire, échanger 2 lignes de façon à avoir $a_{1,1} \neq 0$.
   - On effectue des opérations $L_i \leftarrow L_i + \lambda L_1$ pour i allant de 2 à n de manière à mettre des zéros sur la 1ère colonne, en dessous de la diagonale.

$$\begin{pmatrix} a'_{1,1} & a'_{1,2} & \cdots & a'_{1,n-1} & a'_{1,n} & b'_1 \\ 0 & a'_{2,2} & \cdots & a'_{2,n-1} & a'_{2,n} & b'_2 \\ 0 & a'_{3,2} & \cdots & a'_{3,n-1} & a'_{3,n} & b'_3 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & a'_{n,2} & \cdots & a'_{n,n-1} & a'_{n,n} & b'_n \end{pmatrix}$$

Descente

$$\begin{pmatrix} a'_{1,1} & a'_{1,2} & \cdots & a'_{1,n-1} & a'_{1,n} & b'_1 \\ 0 & a'_{2,2} & \cdots & a'_{2,n-1} & a'_{2,n} & b'_2 \\ 0 & a'_{3,2} & \cdots & a'_{3,n-1} & a'_{3,n} & b'_3 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & a'_{n,2} & \cdots & a'_{n,n-1} & a'_{n,n} & b'_n \end{pmatrix}$$

2 Deuxième étape : on recommence avec la 2ème colonne

- recherche d'un deuxième pivot non nul $a''_{2,2}$
- élimination des termes sous le pivot

Descente

$$\begin{pmatrix} a'_{1,1} & a'_{1,2} & \cdots & a'_{1,n-1} & a'_{1,n} & b'_1 \\ 0 & a'_{2,2} & \cdots & a'_{2,n-1} & a'_{2,n} & b'_2 \\ 0 & a'_{3,2} & \cdots & a'_{3,n-1} & a'_{3,n} & b'_3 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & a'_{n,2} & \cdots & a'_{n,n-1} & a'_{n,n} & b'_n \end{pmatrix}$$

2 Deuxième étape : on recommence avec la 2ème colonne

- recherche d'un deuxième pivot non nul $a''_{2,2}$
- élimination des termes sous le pivot

$$\begin{pmatrix} a'_{1,1} & a'_{1,2} & \cdots & a'_{1,n-1} & a'_{1,n} & b'_1 \\ 0 & a''_{2,2} & \cdots & a''_{2,n-1} & a''_{2,n} & b''_2 \\ 0 & 0 & \cdots & a''_{3,n-1} & a''_{3,n} & b''_3 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & a''_{n,n-1} & a''_{n,n} & b''_n \end{pmatrix}$$

Descente

3 Étapes suivantes : on recommence le processus afin d'obtenir une matrice échelonnée

$$\begin{pmatrix} a'_{1,1} & a'_{1,2} & \cdots & a'_{1,n-1} & a'_{1,n} & b'_1 \\ 0 & a'_{2,2} & \cdots & a'_{2,n-1} & a'_{2,n} & b'_2 \\ 0 & 0 & \cdots & a'_{3,n-1} & a'_{3,n} & b'_3 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & a'_{n,n} & b'_n \end{pmatrix}$$

Descente

3 Étapes suivantes : on recommence le processus afin d'obtenir une matrice échelonnée

$$\begin{pmatrix} a'_{1,1} & a'_{1,2} & \cdots & a'_{1,n-1} & a'_{1,n} & b'_1 \\ 0 & a'_{2,2} & \cdots & a'_{2,n-1} & a'_{2,n} & b'_2 \\ 0 & 0 & \cdots & a'_{3,n-1} & a'_{3,n} & b'_3 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & a'_{n,n} & b'_n \end{pmatrix}$$

Remarque : s'il n'est pas possible de trouver un pivot (non nul) sur la i^{eme} colonne dans les lignes L_i à L_n, la matrice A n'est pas inversible, on interrompt l'algorithme et on retourne un message d'erreur.

Remontée

1. Par multiplication de chaque ligne L_i par $\frac{1}{a'_{i,i}}$, on met des 1 sur la diagonale.

Remontée

1. Par multiplication de chaque ligne L_i par $\frac{1}{a'_{i,i}}$, on met des 1 sur la diagonale.
2. On met des 0 au dessus de la diagonale de la dernière colonne.

$$\begin{pmatrix} 1 & a''_{1,2} & \cdots & a''_{1,n-1} & 0 & b''_1 \\ 0 & 1 & \cdots & a''_{2,n-1} & 0 & b''_2 \\ 0 & 0 & \cdots & a''_{3,n-1} & 0 & b''_3 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & 1 & b''_n \end{pmatrix}$$

3. On fait de même sur les autres colonnes, de droite à gauche.

Remontée

A l'issue de ces opérations, on aboutit à une matrice augmentée échelonnée et réduite du type :

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & b''_1 \\ 0 & 1 & \cdots & 0 & 0 & b''_2 \\ 0 & 0 & \cdots & 0 & 0 & b''_3 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & 1 & b''_n \end{pmatrix}$$

Remontée

A l'issue de ces opérations, on aboutit à une matrice augmentée échelonnée et réduite du type :

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & b_1'' \\ 0 & 1 & \cdots & 0 & 0 & b_2'' \\ 0 & 0 & \cdots & 0 & 0 & b_3'' \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & 1 & b_n'' \end{pmatrix}$$

La solution du système $AX = B$ est alors $X = \begin{pmatrix} b_1'' \\ b_2'' \\ b_3'' \\ \cdots \\ b_n'' \end{pmatrix}$.

Comment déterminer A^{-1} ?

Comment déterminer A^{-1} ?

On peut utiliser la matrice augmentée : $(A|I_n)$

$$(A|I_n) = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} & 1 & 0 & \cdots & 0 \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} & 0 & 0 & \cdots & 1 \end{pmatrix}$$

Comment déterminer A^{-1} ?
On peut utiliser la matrice augmentée : $(A|I_n)$

$$(A|I_n) = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} & 1 & 0 & \cdots & 0 \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} & 0 & 0 & \cdots & 1 \end{pmatrix}$$

Après échelonnement et réduction du système on obtient :

$$(I_n|A^{-1}) = \begin{pmatrix} 1 & 0 & \cdots & 0 & c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ 0 & 1 & \cdots & 0 & c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{pmatrix}$$

On peut finalement extraire $A^{-1} = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{pmatrix}$

Nous allons implémenter la méthode de Gauss en TP.
Les matrices seront des tableaux 2D numpy.

Nous allons implémenter la méthode de Gauss en TP.
Les matrices seront des tableaux 2D numpy.

Exercice

Écrire une fonction `echange` telle que `echange(M, i, j)` échange les lignes i et j de `M` ($L_i \leftrightarrow L_j$).

Exercice

Écrire une fonction `echange` telle que `echange(M, i, j)` échange les lignes i et j de `M`.

1ère possibilité:

```
def echange(M, i, j):
    tmp = M[i].copy()
    M[i] = M[j].copy()
    M[j] = tmp
```

Exercice

Écrire une fonction `echange` telle que `echange(M, i, j)` échange les lignes i et j de `M`.

2ème possibilité:

```
def echange(M, i, j):
    M[i], M[j] = M[j].copy(), M[i].copy()
```

Exercice

Écrire une fonction `dilatation` telle que `dilatation(M, i, a)` réalise l'opération $L_i \leftarrow aL_i$.

Exercice

Écrire une fonction `transvection` telle que `transvection(M, i, j, a)` réalise l'opération $L_i \leftarrow L_i + a \times L_j$.

Exercice

Écrire une fonction `pivot` telle que `pivot(M, j)` renvoie une ligne d'un coefficient non nul sous la diagonale de la `j`ème colonne.

Exercice

Écrire une fonction `echelonner` réalisant la descente du pivot de Gauss.

Complexité de la méthode du pivot de Gauss

On veut résoudre $AX = B$, avec A une matrice $n \times n$.

1. Construire la matrice augmentée
2. Descente: pour toute colonne, de gauche à droite...
   - Trouver un pivot
   - Mettre ce pivot sur la diagonale
   - Mettre des 0 en dessous de la diagonale
3. Remontée: pour toute colonne, de droite à gauche...
   - Mettre des 0 au dessus de la diagonale

Quelle est la complexité de la méthode du pivot de Gauss? On compte les multiplications et additions.

Complexité de la méthode du pivot de Gauss

On veut résoudre $AX = B$, avec A une matrice $n \times n$.

1. Construire la matrice augmentée **$O(n^2)$ car il faut remplir une matrice $n \times (n+1)$**
2. Descente: pour toute colonne, de gauche à droite...
   - Trouver un pivot
   - Mettre ce pivot sur la diagonale
   - Mettre des 0 en dessous de la diagonale
3. Remontée: pour toute colonne, de droite à gauche...
   - Mettre des 0 au dessus de la diagonale

Quelle est la complexité de la méthode du pivot de Gauss?

Complexité de la méthode du pivot de Gauss

On veut résoudre $AX = B$, avec A une matrice $n \times n$.

1. Construire la matrice augmentée $O(n^2)$
2. Descente: pour toute colonne, de gauche à droite...
   - Trouver un pivot **$O(n)$ pour parcourir toutes les lignes en dessous de la diagonale**
   - Mettre ce pivot sur la diagonale
   - Mettre des 0 en dessous de la diagonale
3. Remontée: pour toute colonne, de droite à gauche...
   - Mettre des 0 au dessus de la diagonale

Quelle est la complexité de la méthode du pivot de Gauss?

Complexité de la méthode du pivot de Gauss

On veut résoudre $AX = B$, avec A une matrice $n \times n$.

1. Construire la matrice augmentée $O(n^2)$
2. Descente: pour toute colonne, de gauche à droite...
   - Trouver un pivot $O(n)$
   - Mettre ce pivot sur la diagonale **$O(n)$ pour échanger 2 lignes**
   - Mettre des 0 en dessous de la diagonale
3. Remontée: pour toute colonne, de droite à gauche...
   - Mettre des 0 au dessus de la diagonale

Quelle est la complexité de la méthode du pivot de Gauss?

Complexité de la méthode du pivot de Gauss

On veut résoudre $AX = B$, avec A une matrice $n \times n$.

1. Construire la matrice augmentée $O(n^2)$
2. Descente: pour toute colonne, de gauche à droite...
   - Trouver un pivot $O(n)$
   - Mettre ce pivot sur la diagonale $O(n)$
   - Mettre des 0 en dessous de la diagonale **$O(n^2)$ car il faut faire au plus n transvections**
3. Remontée: pour toute colonne, de droite à gauche...
   - Mettre des 0 au dessus de la diagonale

Quelle est la complexité de la méthode du pivot de Gauss?

Complexité de la méthode du pivot de Gauss

On veut résoudre $AX = B$, avec A une matrice $n \times n$.

1. Construire la matrice augmentée $O(n^2)$
2. Descente en $\boldsymbol{n \times (O(n) + O(n) + O(n^2)) = O(n^3)}$: pour toute colonne, de gauche à droite...
   - Trouver un pivot $O(n)$
   - Mettre ce pivot sur la diagonale $O(n)$
   - Mettre des 0 en dessous de la diagonale $O(n^2)$
3. Remontée: pour toute colonne, de droite à gauche...
   - Mettre des 0 au dessus de la diagonale

Quelle est la complexité de la méthode du pivot de Gauss?

Complexité de la méthode du pivot de Gauss

On veut résoudre $AX = B$, avec A une matrice $n \times n$.

1. Construire la matrice augmentée $O(n^2)$
2. Descente en $O(n^3)$: pour toute colonne, de gauche à droite...
   - Trouver un pivot $O(n)$
   - Mettre ce pivot sur la diagonale $O(n)$
   - Mettre des 0 en dessous de la diagonale $O(n^2)$
3. Remontée: pour toute colonne, de droite à gauche...
   - Mettre des 0 au dessus de la diagonale **$O(n^2)$ car il faut faire au plus n transvections**

Quelle est la complexité de la méthode du pivot de Gauss?

Complexité de la méthode du pivot de Gauss

On veut résoudre $AX = B$, avec A une matrice $n \times n$.

1. Construire la matrice augmentée $O(n^2)$
2. Descente en $O(n^3)$: pour toute colonne, de gauche à droite...
   - Trouver un pivot $O(n)$
   - Mettre ce pivot sur la diagonale $O(n)$
   - Mettre des 0 en dessous de la diagonale $O(n^2)$
3. Remontée en $\boldsymbol{n \times O(n^2)}$: pour toute colonne, de droite à gauche...
   - Mettre des 0 au dessus de la diagonale $\boldsymbol{O(n^2)}$ **car il faut faire au plus** n **transvections**

Quelle est la complexité de la méthode du pivot de Gauss?

Complexité de la méthode du pivot de Gauss

On veut résoudre $AX = B$, avec A une matrice $n \times n$.

1. Construire la matrice augmentée $O(n^2)$
2. Descente en $O(n^3)$: pour toute colonne, de gauche à droite...
   - Trouver un pivot $O(n)$
   - Mettre ce pivot sur la diagonale $O(n)$
   - Mettre des 0 en dessous de la diagonale $O(n^2)$
3. Remontée en $O(n^3)$: pour toute colonne, de droite à gauche...
   - Mettre des 0 au dessus de la diagonale

La complexité totale est donc $\boxed{\boldsymbol{O(n^3)}}$.

Méthode du pivot de Gauss

Conclusion:

1. Il est possible de résoudre un système de n équations à n inconnues en complexité $O(n^3)$.
2. Il est possible d'inverser une matrice inversible $n \times n$ en complexité $O(n^3)$
3. Il est possible de calculer le déterminant d'une matrice $n \times n$ en complexité $O(n^3)$