

ResolSysteme [fr]

Des outils pour des matrices,
des systèmes linéaires,
avec xint ou pyluatex.

Version 0.1.6 -- 2 avril 2024

Cédric Pierquet

c pierquet -- at -- outlook . fr

<https://github.com/cpierquet/ResolSysteme>

- Une commande pour afficher une matrice carrée (2x2, 3x3 ou 4x4) avec la syntaxe du package.
- Quelques commandes pour effectuer des calculs matriciels (produit, carré, puissance).
- Des commandes pour calculer le déterminant et l'inverse de matrices carrées (2x2, 3x3 ou 4x4).
- Des commandes pour résoudre des systèmes linéaires (2x2, 3x3 ou 4x4).
- Des commandes pour travailler sur des graphes probabilistes (2x2, 3x3 ou 4x4).

$\$M=\backslash\text{AffMatrice}(1,2 \text{ § } 3,4)\$,$ et $\$M^3=\backslash\text{MatricePuissancePY}(1,2 \text{ § } 3,4)(3)\$.$

La matrice $M = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ au cube vaut $M^3 = \begin{pmatrix} 37 & 54 \\ 81 & 118 \end{pmatrix}$.

Le **déterminant** de $A = \begin{pmatrix} -1 & \frac{1}{2} \\ \frac{1}{2} & 4 \end{pmatrix}$ est $\det(A) = -4,25$.

L'**inverse** de la matrice $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 0 \\ 1 & 1 & 1 & 1 \\ -2 & -3 & -5 & -6 \end{pmatrix}$ est $A^{-1} = \begin{pmatrix} -15/8 & -1/8 & 3/2 & -1 \\ 23/8 & 1/8 & 1/2 & 2 \\ -9/8 & 1/8 & -3/2 & -1 \\ 1/8 & -1/8 & 1/2 & 0 \end{pmatrix}$.

La **solution** de $\begin{cases} y + z + t = 1 \\ x + z + t = -1 \\ x + y + t = 1 \\ x + y + z = 0 \end{cases}$ est $\mathcal{S} = \left\{ \left(-\frac{2}{3}; \frac{4}{3}; -\frac{2}{3}; \frac{1}{3} \right) \right\}$.

Merci à Denis Bitouzé et à Gilles Le Bourhis pour leurs retours et idées!

LaTeX

pdfLaTeX

LuaLaTeX

TikZ

TeXLive

MiKTeX

Table des matières

I	Introduction	3
1	Le package ResolSysteme	3
1.1	Introduction	3
1.2	Packages utilisés, choix de formatage	3
1.3	Fichiers d'exemples	3
1.4	Chargement du package, et option	4
2	Comparaison avec d'autres solutions	4
II	Historique	4
III	Commandes et calculs matriciels	5
3	Une commande interne : écriture sous forme d'une fraction	5
3.1	La commande	5
3.2	Utilisation	5
3.3	Interaction avec les commandes « matricielles », limitations	5
4	Affichage d'une matrice carrée	6
4.1	La commande	6
4.2	Utilisation	6
5	Calculs matriciels « simples »	7
5.1	Introduction	7
5.2	Utilisation	8
6	Calcul de déterminant	10
6.1	Introduction	10
6.2	Utilisation	10
7	Inverse d'une matrice	12
7.1	Introduction	12
7.2	Utilisation	12
8	États avec un graphe probabiliste	14
8.1	Introduction	14
8.2	Utilisation	14
IV	Résolution de systèmes	16
9	Résolution d'un système linéaire	16
9.1	Introduction	16
9.2	Utilisation	16
10	Recherche d'un état stable (graphe probabiliste)	18
10.1	Introduction	18
10.2	Utilisation	18
V	Fonctions python utilisées	20

Première partie

Introduction

1 Le package ResolSysteme

1.1 Introduction



La package *propose* des outils pour travailler sur des matrices ou des systèmes linéaires ou des graphes probabilistes (de tailles réduites!) :

- en calculant des **produits matriciels simples** (dimensions réduites);
- en affichant la **solution** (si elle existe) d'un système linéaire;
- en affichant le **déterminant** et l'éventuelle **inverse** de la matrice des coefficients;
- en déterminant un **état probabiliste** ou l'éventuel **état stable** d'un graphe probabiliste.



À noter que les calculs – en interne – peuvent être effectués de deux manières :

- via les packages `xint*` pour des formats **2x2** ou **3x3** (et dans une certaine mesure pour des **4x4**);
- via python et le package `pyluatex` (à charger manuellement du fait des options spécifiques) pour des formats **2x2**, **3x3** ou **4x4**.

Il n'est pas prévu – pour le moment – de travailler sur des matrices/systèmes plus grands, car l'idée est de pouvoir formater le résultat, ce qui se fait coefficient par coefficient.



L'utilisation de `pyluatex` nécessite une compilation adaptée, à savoir en `LuaLATEX` et en activant le mode `-shell-escape`.

La méthode par python utilise quoi qu'il en soit le module `sympy`, qui doit donc être installé!

1.2 Packages utilisés, choix de formatage



ResolSysteme charge les packages suivantes :

- `xintexpr`, `xinttools`, `xstring` et `listofitems`;
- `sinuitx`, `nicefrac` et `nicematrix`;

Il est compatible avec les compilations usuelles en `latex`, `pdflatex`, `lualatex` (obligatoire pour `pyluatex`!!) ou `xelatex`.



Les nombres sont formatés par la commande `\num` de `sinuitx`, donc les options choisies par l'utilisateur se propageront aux résultats numériques.

L'affichage des matrices est gérée par le package `nicematrix`, et des options spécifiques *simples* pourront être placées dans les différentes commandes.

1.3 Fichiers d'exemples



En marge de la présente documentation, compilée en `lualatex` avec `shell-escape`, deux fichiers avec des exemples d'utilisation sont proposés :

- `ResolSysteme-exemples` pour les commandes disponibles en version classique (`xint`);
- `ResolSysteme-exemples-pyluatex` pour les commandes disponibles en version python (`pyluatex`).

1.4 Chargement du package, et option



Le package peut donc se charger de deux manières différentes, suivant si l'utilisateur utilise python ou non. Les commandes *classiques* sont disponibles même si python est utilisé.

Code \LaTeX

```
%chargement du package sans passer par pyluatex, calculs via xint  
\usepackage{ResolSysteme}
```

Code \LaTeX

```
%chargement du package pyluatex et du package avec [pyluatex]  
\usepackage[options]{pyluatex}  
\usepackage[pyluatex]{ResolSysteme}
```

2 Comparaison avec d'autres solutions



D'autres solutions existent pour faire du calcul matriciel, on peut par exemple citer les excellents packages `calculator` ou `lualinalg`!

L'idée est ici de proposer une version, adaptée à des dimensions classiques, avec formatage des calculs, sous forme de fraction irréductible notamment. Les formatages étant effectués *a posteriori*, j'ai choisi de limiter ce package à des formats de taille raisonnable (**1x2** à **4x4**).

Deuxième partie

Historique

- v0.1.6 : Correction de bugs dans certains calculs.
- v0.1.5 : Inverse d'une matrice 4x4 et système 4x4 (même en normal).
- v0.1.4 : Ajout de commandes pour du calcul matriciel sans python (de taille raisonnable); commandes pour des graphes probabilistes.
- v0.1.3 : Ajout de commandes pour du calcul matriciel (de taille raisonnable); inversion du comportement des commandes étoilées.
- v0.1.2 : Ajout d'une commande d'affichage (formaté) d'une matrice 2x2, 3x3 ou 4x4.
- v0.1.1 : Correction d'un bug avec le caractère « ; ».
- v0.1.0 : Version initiale.

Troisième partie

Commandes et calculs matriciels

3 Une commande interne : écriture sous forme d'une fraction

3.1 La commande



En *interne*, le code utilise une commande pour formater un résultat sous forme fractionnaire, avec gestion des entiers et gestion du signe « - ».

Code \LaTeX

```
\ConvVersFrac(*)[option de formatage]{calcul}
```

3.2 Utilisation



Concernant cette commande, qui est dans un bloc `ensuremath` :

- `0.1.3` la version *étoilée* force l'écriture du signe « - » sur le numérateur ;
- le premier argument, *optionnel* et entre [...] permet de spécifier un formatage du résultat :
 - `<t>` pour l'affichage de la fraction en mode `tfrac` ;
 - `<d>` pour l'affichage de la fraction en mode `dfrac` ;
 - `<n>` pour l'affichage de la fraction en mode `nicefrac` ;
 - `<dec>` pour l'affichage du résultat en mode décimal (sans arrondi!) ;
 - `<dec=k>` pour l'affichage du résultat en mode décimal arrondi à 10^{-k} ;
- le second argument, *obligatoire*, est quant à lui, le calcul en syntaxe `xint`.

Code \LaTeX

```
\ConvVersFrac{-10+1/3*(-5/16)}           %sortie par défaut
\ConvVersFrac*{-10+1/3*(-5/16)}          %sortie fraction avec - sur numérateur
\ConvVersFrac[d]{-10+1/3*(-5/16)}        %sortie en displaystyle
\ConvVersFrac[n]{-10+1/3*(-5/16)}        %sortie en nicefrac
\ConvVersFrac[dec=4]{-10+1/3*(-5/16)}    %sortie en décimal arrondi à 0,0001
\ConvVersFrac{2+91/7}                    %entier correctement formaté
```

Code \LaTeX

$$-\frac{485}{48} \quad \frac{-485}{48} \quad -\frac{485}{48} \quad -485/48 \quad -10,1042 \quad 15$$

3.3 Interaction avec les commandes « matricielles », limitations



En *interne*, le formatage des résultats est donc géré par cette commande, et les options disponibles existent donc de la même manière pour les commandes liées aux systèmes linéaires et aux calculs matriciels.

Il ne sera par contre pas possible de spécifier des options différentes pour chacun des coefficients, autrement dit l'éventuelle option se propagera sur l'ensemble des résultats !

Les *transformations* en fraction devraient pouvoir fonctionner avec des calculs *classiques*, mais il est possible que, dans des cas *spécifiques*, les résultats ne soient pas ceux attendus !

4 Affichage d'une matrice carrée

4.1 La commande



Une commande (matricielle) est dédiée à l'affichage d'une matrice **2x2** ou **3x3** ou **4x4** (python est ici non nécessaire!) :

- en saisissant les coefficients via une syntaxe propre au package (l'affichage est géré en interne par nicematrix);
- en calculant et convertissant éventuellement les coefficients sous forme de fraction (grâce à la commande précédente!).

Code \LaTeX

```
%commande disponible avec les deux versions, pyluatex ou non  
\AffMatrice(*)[option de formatage]<(matrice)
```

4.2 Utilisation



Concernant cette commande, qui est à insérer dans un environnement *math* :

- `opt 0.1.3` la version *étoilée* force l'écriture du signe « - » sur le numérateur;
- le premier argument, *optionnel* et entre [...] permet de spécifier un formatage du résultat :
 - `<t>` pour l'affichage de la fraction en mode tfrac;
 - `<d>` pour l'affichage de la fraction en mode dfrac;
 - `<n>` pour l'affichage de la fraction en mode nicefrac;
 - `<dec>` pour l'affichage du résultat en mode décimal (sans arrondi!);
 - `<dec=k>` pour l'affichage du résultat en mode décimal arrondi à 10^{-k} ;
- le deuxième argument, *optionnel* et entre <...> correspond aux `<options>` à passer à l'environnement pNiceMatrix;
- le troisième argument, *obligatoire* et entre (...), est quant à lui, la matrice donnée par ses coefficients `a11,a12,... § a21,a22,...` (syntaxe *inspirée* de sympy).

Code \LaTeX

```
On considère les matrices $A=\AffMatrice(1,2 § 3,4)$  
et $B=\AffMatrice[n](-1,1/3,4 § 1/3,4,-1 § -1,0,0)$  
et $C=\AffMatrice(1,2,3,4 § 5,6,7,0 § 1,1,1,1 § 2,-3,-5,-6)$.
```

On considère les matrices $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ et $B = \begin{pmatrix} -1 & 1/3 & 4 \\ 1/3 & 4 & -1 \\ -1 & 0 & 0 \end{pmatrix}$ et $C = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & -3 & -5 & -6 \end{pmatrix}$.

Code \LaTeX

```
On considère la matrice  
$M=\AffMatrice[d]<cell-space-limits=2pt>(1+1/4,0,3+4/5 § 0,1,-5/3 § 1/2,0.45,6/7)$.
```

On considère la matrice $M = \begin{pmatrix} \frac{5}{4} & 0 & \frac{19}{5} \\ 0 & 1 & -\frac{5}{3} \\ \frac{1}{2} & \frac{9}{20} & \frac{6}{7} \end{pmatrix}$.

5 Calculs matriciels « simples »

5.1 Introduction



L'idée est de proposer des commandes pour effectuer des calculs matriciels *simples* sur des matrices :

— des produits matriciels :

- $(1 \times 2) \times (2 \times 1)$;
- $(1 \times 2) \times (2 \times 2)$;
- $(2 \times 2) \times (2 \times 2)$;
- $(2 \times 2) \times (2 \times 1)$;
- $(1 \times 3) \times (3 \times 1)$;
- $(1 \times 3) \times (3 \times 3)$;
- $(3 \times 3) \times (3 \times 3)$;
- $(3 \times 3) \times (3 \times 1)$;
- $(1 \times 4) \times (4 \times 1)$;
- $(1 \times 4) \times (4 \times 4)$;
- $(4 \times 4) \times (4 \times 4)$;
- $(4 \times 4) \times (4 \times 1)$;

— le carré d'une matrice 2x2 ou 3x3 ou 4x4 ;

— la puissance d'une matrice 2x2 ou 3x3 ou 4x4 (via python).

Code \LaTeX

```
%commandes disponible avec les deux versions, pyluatex ou non
\ProduitMatrices(*)[option de formatage]<options nicematrix>(matrice 1)(matrice 2)[Clé]
\ProduitMatricesPY(*)[option de formatage]<options nicematrix>(matrice 1)(matrice 2)[Clé]
\CarreMatrice(*)[option de formatage]<options nicematrix>(matrice)[Clé]

%commande disponible avec l'option pyluatex
\MatricePuissancePY(*)[option de formatage]<options nicematrix>(matrice)(puissance)[Clé]
```



Dans le cas où le produit matriciel n'existe pas (un test de dimensions est effectué), ou ne rentre pas dans le cadre des cas possibles, rien ne sera affiché !

5.2 Utilisation



Concernant ces commandes, qui sont à insérer dans un environnement *math* :

- `\M{0.1.3}` la version *étoilée* force l'écriture du signe « - » sur le numérateur ;
- le premier argument, *optionnel* et entre [...] permet de spécifier un formatage du résultat :
 - `<t>` pour l'affichage de la fraction en mode `tfrac` ;
 - `<d>` pour l'affichage de la fraction en mode `dfrac` ;
 - `<n>` pour l'affichage de la fraction en mode `nicefrac` ;
 - `<dec>` pour l'affichage du résultat en mode décimal (sans arrondi!) ;
 - `<dec=k>` pour l'affichage du résultat en mode décimal arrondi à 10^{-k} ;
- le deuxième argument, *optionnel* et entre `<...>` correspond aux **<options>** à passer à l'environnement `pNiceMatrix` ;
- les arguments suivants, *obligatoires* et entre (...), sont quant à eux, les matrices données par leurs coefficients `a11,a12,... § a21,a22,...` (syntaxe *inspirée* de `sympy`) ou la matrice et la puissance ;
- le dernier argument, *optionnel* et entre [...] propose l'unique « clé » **<Aff>** pour afficher le calcul avant le résultat.

Code \LaTeX

```
\ProduitMatrices(-5,6 § 1,4)(2 § 7)[Aff]$ et \ProduitMatrices(-5,6 § 1,4)(2 § 7)$
```

$$\begin{pmatrix} -5 & 6 \\ 1 & 4 \end{pmatrix} \times \begin{pmatrix} 2 \\ 7 \end{pmatrix} = \begin{pmatrix} 32 \\ 30 \end{pmatrix} \text{ et } \begin{pmatrix} 32 \\ 30 \end{pmatrix}$$

Code \LaTeX

```
\ProduitMatrices[dec](0.5,0.3,0.2)(0.75,0.1,0.15 § 0.4,0.4,0.2 § 0.6,0.1,0.3)[Aff]$
```

$$(0,5 \ 0,3 \ 0,2) \times \begin{pmatrix} 0,75 & 0,1 & 0,15 \\ 0,4 & 0,4 & 0,2 \\ 0,6 & 0,1 & 0,3 \end{pmatrix} = (0,615 \ 0,19 \ 0,195)$$

Code \LaTeX

```
\ProduitMatrices(1,1,1,5 § 2,1,5,6 § 0,5,-6,0 § 1,-5,4,2)(1 § 2 § 3 § 4)[Aff]$
```

$$\begin{pmatrix} 1 & 1 & 1 & 5 \\ 2 & 1 & 5 & 6 \\ 0 & 5 & -6 & 0 \\ 1 & -5 & 4 & 2 \end{pmatrix} \times \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 26 \\ 43 \\ -8 \\ 11 \end{pmatrix}$$

Code \LaTeX

```
\ProduitMatrices%
(1,1,1,5 § 2,1,5,6 § 0,5,-6,0 § 1,-5,4,2)%
(1,5,4,0 § 2,-1,-1,5 § 3,0,1,2, § 4,6,9,10)
[Aff]$
```

$$\begin{pmatrix} 1 & 1 & 1 & 5 \\ 2 & 1 & 5 & 6 \\ 0 & 5 & -6 & 0 \\ 1 & -5 & 4 & 2 \end{pmatrix} \times \begin{pmatrix} 1 & 5 & 4 & 0 \\ 2 & -1 & -1 & 5 \\ 3 & 0 & 1 & 2 \\ 4 & 6 & 9 & 10 \end{pmatrix} = \begin{pmatrix} 26 & 34 & 49 & 57 \\ 43 & 45 & 66 & 75 \\ -8 & -5 & -11 & 13 \\ 11 & 22 & 31 & 3 \end{pmatrix}$$

Code \LaTeX

```
\CarreMatrice(-5,6 § 1,4)[Aff]
```

$$\begin{pmatrix} -5 & 6 \\ 1 & 4 \end{pmatrix}^2 = \begin{pmatrix} 31 & -6 \\ -1 & 22 \end{pmatrix}$$

Code \LaTeX

```
\CarreMatrice(-5,6,8 § 1,4,-9 § 1,-1,1)[Aff]
```

$$\begin{pmatrix} -5 & 6 & 8 \\ 1 & 4 & -9 \\ 1 & -1 & 1 \end{pmatrix}^2 = \begin{pmatrix} 39 & -14 & -86 \\ -10 & 31 & -37 \\ -5 & 1 & 18 \end{pmatrix}$$

Code \LaTeX

```
\MatricePuissancePY(1,1 § 5,-2)(7)[Aff]
```

$$\begin{pmatrix} 1 & 1 \\ 5 & -2 \end{pmatrix}^7 = \begin{pmatrix} -559 & 673 \\ 3365 & -2578 \end{pmatrix}$$

Code \LaTeX

```
\MatricePuissancePY(1,1,-1 § 5,-2,1 § 0,5,2)(3)[Aff]
```

$$\begin{pmatrix} 1 & 1 & -1 \\ 5 & -2 & 1 \\ 0 & 5 & 2 \end{pmatrix}^3 = \begin{pmatrix} -24 & 8 & -16 \\ 65 & -58 & 9 \\ 25 & 70 & -7 \end{pmatrix}$$

Code \LaTeX

```
\MatricePuissancePY(1,1,1,1 § 5,-2,1,5 § 0,5,2,-1 § 0,1,1,1)(5)[Aff]
```

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 5 & -2 & 1 & 5 \\ 0 & 5 & 2 & -1 \\ 0 & 1 & 1 & 1 \end{pmatrix}^5 = \begin{pmatrix} 886 & 769 & 769 & 913 \\ 1730 & 847 & 1090 & 1655 \\ 1395 & 1865 & 1622 & 1565 \\ 720 & 625 & 625 & 742 \end{pmatrix}$$

6 Calcul de déterminant

6.1 Introduction



Une commande est disponible pour calculer le déterminant d'une matrice :

- **2x2** ou **3x3** ou **4x4**.

Code \LaTeX

```
%version classique
\DetMatrice(*)[option de formatage](matrice)

%version python
\DetMatricePY(*)[option de formatage](matrice)
```

6.2 Utilisation



Concernant cette commande, qui est à insérer dans un environnement *math* :

- `\det` la version *étoilée* force l'écriture du signe « - » sur le numérateur ;
- le premier argument, *optionnel* et entre [...] permet de spécifier un formatage du résultat :
 - `<t>` pour l'affichage de la fraction en mode tfrac ;
 - `<d>` pour l'affichage de la fraction en mode dfrac ;
 - `<n>` pour l'affichage de la fraction en mode nicefrac ;
 - `<dec>` pour l'affichage du résultat en mode décimal (sans arrondi!) ;
 - `<dec=k>` pour l'affichage du résultat en mode décimal arrondi à 10^{-k} ;
- le second argument, *obligatoire* et entre (...), est quant à lui, la matrice donnée par ses coefficients $a_{11}, a_{12}, \dots, a_{21}, a_{22}, \dots$ (syntaxe *inspirée* de sympy).

Code \LaTeX

```
%version classique
Le dét. de $A=\AffMatrice(1,2 § 3,4)$ est
$\det(A)=\DetMatrice(1,2 § 3,4)$.
```

Le dét. de $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ est $\det(A) = -2$.

Code \LaTeX

```
%version classique
Le dét. de $A=\AffMatrice[dec](-1,0.5 § 1/2,4)$ est
$\det(A)=\DetMatrice[dec](-1,0.5 § 1/2,4)$.
```

Le dét. de $A = \begin{pmatrix} -1 & 0,5 \\ 0,5 & 4 \end{pmatrix}$ est $\det(A) = -4,25$.

Code \LaTeX

```
%version classique
Le dét. de $A=\AffMatrice[t](-1,1/3,4 § -1/3,4,-1 § -1,0,0)$ est
$\det(A) \approx \DetMatrice[dec=3](-1,1/3,4 § -1/3,4,-1 § -1,0,0)$.
```

Le dét. de $A = \begin{pmatrix} -1 & \frac{1}{3} & 4 \\ -\frac{1}{3} & 4 & -1 \\ -1 & 0 & 0 \end{pmatrix}$ est $\det(A) \approx 16,333$.

Le dét. de $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & -3 & -5 & -6 \end{pmatrix}$ est $\det(A) = \text{DetMatrice}(1,2,3,4 \text{ } 5,6,7,0 \text{ } 1,1,1,1 \text{ } 2,-3,-5,-6)$.

Le dét. de $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & -3 & -5 & -6 \end{pmatrix}$ est $\det(A) = 24$.

```
%version python
Le dé. de  $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$  est
 $\det(A) = \text{DetMatricePY}(1,2 \text{ } 3,4)$ .
```

Le dé. de $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ est $\det(A) = -2$.

```
Le dét. de  $A = \begin{pmatrix} -1 & 0,5 \\ 0,5 & 4 \end{pmatrix}$  est
 $\det(A) = \text{DetMatricePY}[d](-1,0.5 \text{ } 1/2,4)$ .
```

Le dét. de $A = \begin{pmatrix} -1 & 0,5 \\ 0,5 & 4 \end{pmatrix}$ est $\det(A) = -\frac{17}{4}$.

```
%version python
Le dét. de  $A = \begin{pmatrix} -1 & 1/3 & 4 \\ 1/3 & 4 & -1 \\ -1 & 0 & 0 \end{pmatrix}$  est
 $\det(A) \approx \text{DetMatricePY}[dec=3](-1,1/3,4 \text{ } 1/3,4,-1 \text{ } -1,0,0)$ .
```

Le dét. de $A = \begin{pmatrix} -1 & \frac{1}{3} & 4 \\ \frac{1}{3} & 4 & -1 \\ -1 & 0 & 0 \end{pmatrix}$ est $\det(A) \approx 16,333$.

```
%version python
Le dét. de  $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & -3 & -5 & -6 \end{pmatrix}$ 
est  $\det(A) = \text{DetMatricePY}(1,2,3,4 \text{ } 5,6,7,0 \text{ } 1,1,1,1 \text{ } 2,-3,-5,-6)$ .
```

Le dét. de $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & -3 & -5 & -6 \end{pmatrix}$ est $\det(A) = 24$.

7 Inverse d'une matrice

7.1 Introduction



Une commande (matricielle) disponible est pour calculer l'éventuelle inverse d'une matrice :

- **2x2** ou **3x3** ou **4x4** (`\Mat 0.1.5`) pour le package *classique* ;
- **2x2** ou **3x3** ou **4x4** également pour la version python.

Code \LaTeX

```
%version classique
\MatriceInverse(*)[option de formatage]<options nicematrix>(matrice) [Clé]

%version python
\MatriceInversePY(*)[option de formatage]<options nicematrix>(matrice) [Clé]
```

7.2 Utilisation



Concernant cette commande, qui est à insérer dans un environnement *math* :

- `\Mat 0.1.3` la version *étoilée* force l'écriture du signe « - » sur le numérateur ;
- le premier argument, *optionnel* et entre [...] permet de spécifier un formatage du résultat :
 - **<t>** pour l'affichage de la fraction en mode tfrac ;
 - **<d>** pour l'affichage de la fraction en mode dfrac ;
 - **<n>** pour l'affichage de la fraction en mode nicefrac ;
 - **<dec>** pour l'affichage du résultat en mode décimal (sans arrondi!) ;
 - **<dec=k>** pour l'affichage du résultat en mode décimal arrondi à 10^{-k} ;
- le deuxième argument, *optionnel* et entre <...> correspond aux **<options>** à passer à l'environnement pNiceMatrix ;
- le troisième argument, *obligatoire* et entre (...), est quant à lui, la matrice donnée par ses coefficients $a_{11}, a_{12}, \dots \ S \ a_{21}, a_{22}, \dots$ (syntaxe *inspirée* de sympy) ;
- le dernier argument, *optionnel* et entre [...] propose l'unique « clé » **<Aff>** pour afficher le calcul avant le résultat.

À noter que si la matrice n'est pas inversible, le texte `Matrice non inversible` est affiché.

Code \LaTeX

```
%version classique
L'inverse de $A=\AffMatrice(1,2 \ S \ 3,4)$ est
$A^{-1}=\MatriceInverse<cell-space-limits=2pt>(1,2 \ S \ 3,4)$.
```

L'inverse de $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ est $A^{-1} = \begin{pmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{pmatrix}$.

Code \LaTeX

```
%version classique
L'inverse de $A=\AffMatrice(1,2,3 \ S \ 4,5,6 \ S \ 7,8,8)$ est
$A^{-1}=\MatriceInverse[n]<cell-space-limits=2pt>(1,2,3 \ S \ 4,5,6 \ S \ 7,8,8)[Aff]$.
```

L'inverse de $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 8 \end{pmatrix}$ est $A^{-1} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 8 \end{pmatrix}^{-1} = \begin{pmatrix} -8/3 & 8/3 & -1 \\ 10/3 & -13/3 & 2 \\ -1 & 2 & -1 \end{pmatrix}$.

```
%version python
L'inverse de $A=\text{AffMatrice}(1,2 \text{ § } 3,4)$ est
$A^{-1}=\text{MatriceInversePY}[d]<\text{cell-space-limits}=2\text{pt}>(1,2 \text{ § } 3,4)[\text{Aff}]$.
```

L'inverse de $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ est $A^{-1} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}^{-1} = \begin{pmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{pmatrix}$.

```
%version normale
L'inv. de $A=\text{AffMatrice}(1,2,3,4 \text{ § } 5,6,7,0 \text{ § } 1,1,1,1 \text{ § } -2,-3,-5,-6)$ est
$A^{-1}=$
\MatriceInverse[n]<\text{cell-space-limits}=2\text{pt}>(1,2,3,4 \text{ § } 5,6,7,0 \text{ § } 1,1,1,1 \text{ § } -2,-3,-5,-6)$.
```

L'inv. de $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 0 \\ 1 & 1 & 1 & 1 \\ -2 & -3 & -5 & -6 \end{pmatrix}$ est $A^{-1} = \begin{pmatrix} -15/8 & -1/8 & 3/2 & -1 \\ 23/8 & 1/8 & 1/2 & 2 \\ -9/8 & 1/8 & -3/2 & -1 \\ 1/8 & -1/8 & 1/2 & 0 \end{pmatrix}$.

```
%version python
L'inv. de $A=\text{AffMatrice}(1,2,3,4 \text{ § } 5,6,7,0 \text{ § } 1,1,1,1 \text{ § } -2,-3,-5,-6)$ est
$A^{-1}=$
\MatriceInversePY[n]<\text{cell-space-limits}=2\text{pt}>(1,2,3,4 \text{ § } 5,6,7,0 \text{ § } 1,1,1,1 \text{ § } -2,-3,-5,-6)$.
```

L'inv. de $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 0 \\ 1 & 1 & 1 & 1 \\ -2 & -3 & -5 & -6 \end{pmatrix}$ est $A^{-1} = \begin{pmatrix} -15/8 & -1/8 & 3/2 & -1 \\ 23/8 & 1/8 & 1/2 & 2 \\ -9/8 & 1/8 & -3/2 & -1 \\ 1/8 & -1/8 & 1/2 & 0 \end{pmatrix}$.

8 États avec un graphe probabiliste

8.1 Introduction



0.1.4 Il existe des commandes pour travailler sur un graphe probabiliste (avec le package en version python) :

- afficher un état probabiliste (**1x2** ou **1x3** ou **1x4**, version normale ou version python);
- déterminer un état probabiliste à une certaine étape, uniquement en version python.

Code \LaTeX

```
%version classique ou python
\AffEtatProb[opt de formatage]<opts nicematrix>(matrice ligne)
\EtatProbPY[opt de formatage]<opts nicematrix>(état init)(mat de trans)(étape)
```

8.2 Utilisation



Concernant la commande d'affichage d'un état, qui est à insérer dans un environnement *math* :

- le premier argument, *optionnel* et entre [...] permet de spécifier un formatage du résultat :
 - **<t>** pour l'affichage de la fraction en mode tfrac;
 - **<d>** pour l'affichage de la fraction en mode dfrac;
 - **<n>** pour l'affichage de la fraction en mode nicefrac;
 - **<dec>** pour l'affichage du résultat en mode décimal (sans arrondi!);
 - **<dec=k>** pour l'affichage du résultat en mode décimal arrondi à 10^{-k} ;
- le deuxième argument, *optionnel* et entre <...> correspond aux **<options>** à passer à l'environnement `pNiceMatrix`;
- le troisième argument, *obligatoire* et entre (...), est quant à lui, la matrice donnée par ses coefficients a_{11}, a_{12}, \dots (syntaxe *inspirée* de `sympy`).



Concernant la commande d'affichage d'un état à une étape donnée, qui est à insérer dans un environnement *math* :

- le premier argument, *optionnel* et entre [...] permet de spécifier un formatage du résultat [dec] par défaut :
 - **<t>** pour l'affichage de la fraction en mode tfrac;
 - **<d>** pour l'affichage de la fraction en mode dfrac;
 - **<n>** pour l'affichage de la fraction en mode nicefrac;
 - **<dec>** pour l'affichage du résultat en mode décimal (sans arrondi!);
 - **<dec=k>** pour l'affichage du résultat en mode décimal arrondi à 10^{-k} ;
- le deuxième argument, *optionnel* et entre <...> correspond aux **<options>** à passer à l'environnement `pNiceMatrix`;
- le troisième argument, *obligatoire* et entre (...), est quant à lui, la matrice donnée par ses coefficients a_{11}, a_{12}, a_{13} (syntaxe *inspirée* de `sympy`);
- le quatrième argument, *obligatoire* et entre (...), est quant à lui, la matrice de transition donnée par ses coefficients a_{11}, a_{12}, \dots § a_{21}, a_{22}, \dots (syntaxe *inspirée* de `sympy`);
- le cinquième argument, *obligatoire* et entre (...), est quant à lui, le numéro de l'étape voulue

État initial : $P_0 = \text{\AffEtatProb}[t](1/3,2/3)\$.$

Matrice de transition :
 $M = \text{\AffMatrice}[dec](0.75,0.25 \S 0.9,0.1)\$$

État à l'instant 5 :
 $P_5 \text{\approx} \text{\EtatProbPY}[dec=3](1/3,2/3)\%$
 $(0.75,0.25 \S 0.9,0.1)$
 $(5)\$$

État initial : $P_0 = \begin{pmatrix} \frac{1}{3} & \frac{2}{3} \end{pmatrix}.$

Matrice de transition : $M = \begin{pmatrix} 0,75 & 0,25 \\ 0,9 & 0,1 \end{pmatrix}$

État à l'instant 5 : $P_5 \approx (0,783 \quad 0,217)$

État initial : $P_0 = \text{\AffEtatProb}[dec](0.33,0.52,0.15)\$.$

Matrice de transition :
 $M = \text{\AffMatrice}[dec]\%$
 $(0.1,0.2,0.7 \S 0.25,0.25,0.5 \S 0.15,0.75,0.1)\$$

État à l'instant 7 :
 $P_7 \text{\approx} \text{\EtatProbPY}[dec=3]$
 $(0.33,0.52,0.15)\%$
 $(0.1,0.2,0.7 \S 0.25,0.25,0.5 \S 0.15,0.75,0.1)$
 $(7)\$$

État initial : $P_0 = (0,33 \quad 0,52 \quad 0,15).$

Matrice de transition : $M = \begin{pmatrix} 0,1 & 0,2 & 0,7 \\ 0,25 & 0,25 & 0,5 \\ 0,15 & 0,75 & 0,1 \end{pmatrix}$

État à l'instant 7 : $P_7 \approx (0,184 \quad 0,432 \quad 0,384)$

État initial : $P_0 = \text{\AffEtatProb}[dec](0.33,0.52,0.15,0)\$.$

Matrice de transition :
 $M = \text{\AffMatrice}[dec]\%$
 $(0.1,0.2,0.3,0.4 \S 0.25,0.25,0.25,0.25 \S 0.15,0.15,0.2,0.5 \S 0.3,0.3,0.2,0.2)\$$

État à l'instant 4 :
 $P_4 \text{\approx} \text{\EtatProbPY}[dec=3]$
 $(0.33,0.52,0.15,0)\%$
 $(0.1,0.2,0.3,0.4 \S 0.25,0.25,0.25,0.25 \S 0.15,0.15,0.2,0.5 \S 0.3,0.3,0.2,0.2)\%$
 $(4)\$$

État initial : $P_0 = (0,33 \quad 0,52 \quad 0,15 \quad 0).$

Matrice de transition : $M = \begin{pmatrix} 0,1 & 0,2 & 0,3 & 0,4 \\ 0,25 & 0,25 & 0,25 & 0,25 \\ 0,15 & 0,15 & 0,2 & 0,5 \\ 0,3 & 0,3 & 0,2 & 0,2 \end{pmatrix}$

État à l'instant 4 : $P_4 \approx (0,211 \quad 0,232 \quad 0,233 \quad 0,324)$

Quatrième partie

Résolution de systèmes

9 Résolution d'un système linéaire

9.1 Introduction



Il existe une commande (matricielle) pour déterminer l'éventuelle solution d'un système linéaire qui s'écrit matriciellement $A \times X = B$:

- **2x2** ou **3x3** ou **4x4** (mat 0.1.5) pour le package *classique* ;
- **2x2** ou **3x3** ou **4x4** également pour le package en version python.

Code \LaTeX

```
%version classique
\SolutionSysteme(*)[opt de formatage]<opts nicematrix>(matriceA)(matriceB) [Clé]

%version python
\SolutionSystemePY(*)[opt de formatage]<opts nicematrix>(matriceA)(matriceB) [Clé]
```

9.2 Utilisation



Concernant cette commande, qui est à insérer dans un environnement *math* :

- mat 0.1.3 la version *étoilée* force l'écriture du signe « - » sur le numérateur ;
- le premier argument, *optionnel* et entre [...] permet de spécifier un formatage du résultat :
 - **<t>** pour l'affichage de la fraction en mode tfrac ;
 - **<d>** pour l'affichage de la fraction en mode dfrac ;
 - **<n>** pour l'affichage de la fraction en mode nicefrac ;
 - **<dec>** pour l'affichage du résultat en mode décimal (sans arrondi !);
 - **<dec=k>** pour l'affichage du résultat en mode décimal arrondi à 10^{-k} ;
- le deuxième argument, *optionnel* et entre <...> correspond aux **<options>** à passer à l'environnement pNiceMatrix ;
- le troisième argument, *obligatoire* et entre (...), est quant à lui, la matrice A donnée par ses coefficients $a_{11}, a_{12}, \dots \S a_{21}, a_{22}, \dots$ (syntaxe *inspirée* de sympy) ;
- le quatrième argument, *obligatoire* et entre (...), est quant à lui, la matrice B donnée par ses coefficients b_{11}, b_{21}, \dots (syntaxe *inspirée* de sympy) ;
- le dernier argument, *optionnel* et entre [...], permet – grâce à la clé **(Matrice)** – de présenter le vecteur solution.

À noter que si la matrice n'est pas inversible, le texte Matrice non inversible est affiché.

Code \LaTeX

```
%version classique
La solution de $\systeme{3x+y-2z=-1,2x-y+z=4,x-y-2z=5}$ est $\mathcal{S}=%
\left\lbrace \SolutionSysteme[d](3,1,-2 \S 2,-1,1 \S 1,-1,-2)(-1,4,5) \right\rbrace.\backslash
```

La solution de
$$\begin{cases} 3x + y - 2z = -1 \\ 2x - y + z = 4 \\ x - y - 2z = 5 \end{cases}$$
 est $\mathcal{S} = \left\{ \left(\frac{1}{2}; -\frac{7}{2}; -\frac{1}{2} \right) \right\}$.


```
%version python
La solution de  $\backslash\text{systeme}\{x+y+z=-1,3x+2y-z=6,-x-y+2z=-5\}$  est  $\backslash\text{mathcal}\{S\}=\%$ 
 $\backslash\text{left}\backslash\text{lbrace } \backslash\text{SolutionSystemePY}(1,1,1 \text{ } \S \text{ } 3,2,-1 \text{ } \S \text{ } -1,-1,2)(-1,6,-5) \backslash\text{right}\backslash\text{rbrace}$ .
```

$$\text{La solution de } \begin{cases} x + y + z = -1 \\ 3x + 2y - z = 6 \\ -x - y + 2z = -5 \end{cases} \text{ est } \mathcal{S} = \{(2; -1; -2)\}.$$

```
%version normal
La solution de  $\backslash\text{systeme}[xyzt]\{x+2y+3z+4t=-10,5x+6y+7z=0,x+y+z+t=4,-2x-3y-5z-6t=7\}$ 
est  $\backslash\text{mathcal}\{S\}=\%$ 
 $\backslash\text{left}\backslash\text{lbrace } \backslash\text{SolutionSysteme}\%$ 
 $\backslash\text{dec}\langle\text{cell-space-limits}=2\text{pt}\rangle\%$ 
 $(1,2,3,4 \text{ } \S \text{ } 5,6,7,0 \text{ } \S \text{ } 1,1,1,1 \text{ } \S \text{ } -2,-3,-5,-6)(-10,0,4,7)\%$ 
 $\backslash\text{right}\backslash\text{rbrace}$ .
```

$$\text{La solution de } \begin{cases} x + 2y + 3z + 4t = -10 \\ 5x + 6y + 7z = 0 \\ x + y + z + t = 4 \\ -2x - 3y - 5z - 6t = 7 \end{cases} \text{ est } \mathcal{S} = \{(17,75; -12,75; -1,75; 0,75)\}.$$

```
%version python
La solution de  $\backslash\text{systeme}[xyzt]\{x+2y+3z+4t=-10,5x+6y+7z=0,x+y+z+t=4,-2x-3y-5z-6t=7\}$ 
est  $\backslash\text{mathcal}\{S\}=\%$ 
 $\backslash\text{left}\backslash\text{lbrace } \backslash\text{SolutionSystemePY}\%$ 
 $\backslash\text{dec}\langle\text{cell-space-limits}=2\text{pt}\rangle\%$ 
 $(1,2,3,4 \text{ } \S \text{ } 5,6,7,0 \text{ } \S \text{ } 1,1,1,1 \text{ } \S \text{ } -2,-3,-5,-6)(-10,0,4,7)\%$ 
 $\backslash\text{right}\backslash\text{rbrace}$ .
```

$$\text{La solution de } \begin{cases} x + 2y + 3z + 4t = -10 \\ 5x + 6y + 7z = 0 \\ x + y + z + t = 4 \\ -2x - 3y - 5z - 6t = 7 \end{cases} \text{ est } \mathcal{S} = \{(17,75; -12,75; -1,75; 0,75)\}.$$

```
%pas de solution
La solution de  $\backslash\text{systeme}\{x+2y=-5,4x+8y=1\}$  est  $\backslash\text{mathcal}\{S\}=\%$ 
 $\backslash\text{left}\backslash\text{lbrace } \backslash\text{SolutionSystemePY}(1,2 \text{ } \S \text{ } 4,8)(-5,1) \backslash\text{right}\backslash\text{rbrace}$ .
```

$$\text{La solution de } \begin{cases} x + 2y = -5 \\ 4x + 8y = 1 \end{cases} \text{ est } \mathcal{S} = \{\text{Matrice non inversible}\}.$$

10 Recherche d'un état stable (graphe probabiliste)

10.1 Introduction



0.1.4 Il existe une commande (matricielle) pour déterminer l'éventuel état stable d'un graphe probabiliste :

- **2x2** pour le package *classique* ;
- **2x2** ou **3x3** ou **4x4** pour le package en version python.

Code \LaTeX

```
%version classique
\EtatStable[opt de formatage]<opts nicematrix>(matriceA)

%version python
\EtatStablePY[opt de formatage]<opts nicematrix>(matriceA)
```

10.2 Utilisation



Concernant cette commande, qui est à insérer dans un environnement *math* :

- le premier argument, *optionnel* et entre [...] permet de spécifier un formatage du résultat :
 - **<t>** pour l'affichage de la fraction en mode tfrac ;
 - **<d>** pour l'affichage de la fraction en mode dfrac ;
 - **<n>** pour l'affichage de la fraction en mode nicefrac ;
 - **<dec>** pour l'affichage du résultat en mode décimal (sans arrondi!) ;
 - **<dec=k>** pour l'affichage du résultat en mode décimal arrondi à 10^{-k} ;
- le deuxième argument, *optionnel* et entre <...> correspond aux **<options>** à passer à l'environnement pNiceMatrix ;
- le troisième argument, *obligatoire* et entre (...), est quant à lui, la matrice donnée par ses coefficients $a_{11}, a_{12}, \dots \S a_{21}, a_{22}, \dots$ (syntaxe *inspirée* de sympy).

Code \LaTeX

```
%version classique
L'état stable du gr. prob. de matrice
\M=\AffMatrice[dec](0.72,0.28 § 0.12,0.88)$

est $\Pi = \EtatStable[d](0.72,0.28 § 0.12,0.88)$
ou $\Pi = \EtatStable[dec](0.72,0.28 § 0.12,0.88)$.
```

L'état stable du gr. prob. de matrice $M = \begin{pmatrix} 0,72 & 0,28 \\ 0,12 & 0,88 \end{pmatrix}$
est $\Pi = \left(\frac{3}{10} \quad \frac{7}{10} \right)$ ou $\Pi = (0,3 \quad 0,7)$.

```
%version python
L'état stable du gr. prob. de matrice
$M=\AffMatrice[dec](0.72,0.28 § 0.12,0.88)$

est $\Pi = \EtatStablePY[d](0.72,0.28 § 0.12,0.88)$
ou $\Pi = \EtatStablePY[dec](0.72,0.28 § 0.12,0.88)$.
```

L'état stable du gr. prob. de matrice $M = \begin{pmatrix} 0,72 & 0,28 \\ 0,12 & 0,88 \end{pmatrix}$
est $\Pi = \left(\frac{3}{10} \quad \frac{7}{10} \right)$ ou $\Pi = (0,3 \quad 0,7)$.

```
%version python
L'état stable du gr. prob. de matrice
$M=\AffMatrice[dec](0.9,0.03,0.07 § 0.30,0.43,0.27 § 0.14,0.07,0.79)$

est $\Pi = \EtatStablePY[d](0.9,0.03,0.07 § 0.30,0.43,0.27 § 0.14,0.07,0.79)$
ou $\Pi = \EtatStablePY[dec](0.9,0.03,0.07 § 0.30,0.43,0.27 § 0.14,0.07,0.79)$.
```

L'état stable du gr. prob. de matrice $M = \begin{pmatrix} 0,9 & 0,03 & 0,07 \\ 0,3 & 0,43 & 0,27 \\ 0,14 & 0,07 & 0,79 \end{pmatrix}$
est $\Pi = \left(\frac{63}{100} \quad \frac{7}{100} \quad \frac{3}{10} \right)$ ou $\Pi = (0,63 \quad 0,07 \quad 0,3)$.

```
%version python
L'état stable du gr. prob. de matrice
$M=\AffMatrice[dec]%
(0.1,0.2,0.3,0.4 § 0.25,0.25,0.25,0.25 § 0.15,0.15,0.2,0.5 § 0.3,0.3,0.2,0.2)$

est $\Pi \approx
\EtatStablePY[dec=5]%
(0.1,0.2,0.3,0.4 § 0.25,0.25,0.25,0.25 § 0.15,0.15,0.2,0.5 § 0.3,0.3,0.2,0.2)$.
```

L'état stable du gr. prob. de matrice $M = \begin{pmatrix} 0,1 & 0,2 & 0,3 & 0,4 \\ 0,25 & 0,25 & 0,25 & 0,25 \\ 0,15 & 0,15 & 0,2 & 0,5 \\ 0,3 & 0,3 & 0,2 & 0,2 \end{pmatrix}$
est $\Pi \approx (0,21123 \quad 0,23235 \quad 0,23274 \quad 0,32368)$.

Cinquième partie

Fonctions python utilisées



Les fonctions utilisées par les packages pyluatex ou pythontex sont données ci-dessous. Elles sont accessibles en *natif* une fois l'option pyluatex activée, grâce notamment à la macro `\py.`

```
#variables symboliques (pour du 4x4 maxi)
import sympy as sy
x = sy.Symbol('x')
y = sy.Symbol('y')
z = sy.Symbol('z')
t = sy.Symbol('t')
```

Code Python

```
#résolution de systèmes
def resol_systeme_QQ(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,u) :
    solution=sy.solve([a*x+b*y+c*z+d*t-e,f*x+g*y+h*z+i*t-j,k*x+l*y+m*z+n*t-o,p*x+q*y+r*z+s*t-u],[x,y,z,t])
    return solution

def resol_systeme_TT(a,b,c,d,e,f,g,h,i,j,k,l) :
    solution=sy.solve([a*x+b*y+c*z-d,e*x+f*y+g*z-h,i*x+j*y+k*z-l],[x,y,z])
    return solution

def resol_systeme_DD(a,b,c,d,e,f) :
    solution=sy.solve([a*x+b*y-c,d*x+e*y-f],[x,y])
    return solution
```

Code Python

```
#déterminant d'une matrice
def det_matrice_QQ(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p) :
    MatTmp = sy.Matrix([[a,b,c,d],[e,f,g,h],[i,j,k,l],[m,n,o,p]])
    DetMatTmp = MatTmp.det()
    return DetMatTmp

def det_matrice_TT(a,b,c,d,e,f,g,h,i) :
    MatTmp = sy.Matrix([[a,b,c],[d,e,f],[g,h,i]])
    DetMatTmp = MatTmp.det()
    return DetMatTmp

def det_matrice_DD(a,b,c,d) :
    MatTmp = sy.Matrix([[a,b],[c,d]])
    DetMatTmp = MatTmp.det()
    return DetMatTmp
```

Code Python

```
#inverse d'une matrice
def inverse_matrice_QQ(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p) :
    MatTmp = sy.Matrix([[a,b,c,d],[e,f,g,h],[i,j,k,l],[m,n,o,p]])
    DetMatTmp = MatTmp.inv()
    return DetMatTmp

def inverse_matrice_DD(a,b,c,d) :
    MatTmp = sy.Matrix([[a,b],[c,d]])
    InvMatTmp = MatTmp.inv()
    return InvMatTmp

def inverse_matrice_TT(a,b,c,d,e,f,g,h,i) :
    MatTmp = sy.Matrix([[a,b,c],[d,e,f],[g,h,i]])
    InvMatTmp = MatTmp.inv()
    return InvMatTmp
```

Code Python

```
#puissance d'une matrice
def puissance_matrice_QQ(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,puiss) :
    MatTmp = sy.Matrix([[a,b,c,d],[e,f,g,h],[i,j,k,l],[m,n,o,p]])
    PuissMatTmp = MatTmp**puiss
    return PuissMatTmp

def puissance_matrice_TT(a,b,c,d,e,f,g,h,i,puiss) :
    MatTmp = sy.Matrix([[a,b,c],[d,e,f],[g,h,i]])
    PuissMatTmp = MatTmp**puiss
    return PuissMatTmp

def puissance_matrice_DD(a,b,c,d,puiss) :
    MatTmp = sy.Matrix([[a,b],[c,d]])
    PuissMatTmp = MatTmp**puiss
    return PuissMatTmp
```

```
def etat_prob_QQ(AA,BB,CC,DD,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,puiss) :
    MatTmpInit = sy.Matrix([[AA,BB,CC,DD]]).T
    MatTmpTrans = sy.Matrix([[a,b,c,d],[e,f,g,h],[i,j,k,l],[m,n,o,p]])
    EtatProbRes = MatTmpInit * MatTmpTrans**puiss
    return EtatProbRes

def etat_prob_TT(AA,BB,CC,a,b,c,d,e,f,g,h,i,puiss) :
    MatTmpInit = sy.Matrix([[AA,BB,CC]]).T
    MatTmpTrans = sy.Matrix([[a,b,c],[d,e,f],[g,h,i]])
    EtatProbRes = MatTmpInit * MatTmpTrans**puiss
    return EtatProbRes

def etat_prob_DD(AA,BB,a,b,c,d,puiss) :
    MatTmpInit = sy.Matrix([[AA,BB]]).T
    MatTmpTrans = sy.Matrix([[a,b],[c,d]])
    EtatProbRes = MatTmpInit * MatTmpTrans**puiss
    return EtatProbRes
```

```
def resol_etat_stable_TT(a,b,c,d,e,f,g,h,i) :
    solution=sy.solve([(a-1)*x+d*y+g*z,b*x+(e-1)*y+h*z,c*x+f*y+(i-1)*z,x+y+z-1],[x,y,z])
    return solution

def resol_etat_stable_QQ(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p) :
    solution=sy.solve([(a-1)*x+e*y+i*z+m*t,b*x+(f-1)*y+j*z+n*t,
    c*x+g*y+(k-1)*z+o*t,d*x+h*y+l*z+(p-1)*t,x+y+z+t-1],[x,y,z,t])
    return solution
```