ResolSysteme [fr]

Des outils pour des matrices,

des systèmes linéaires,

avec xint ou pyluatex.

Version 0.1.3 -- 9 Février 2023

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).

```
$M=\AffMatrice(1,2 § 3,4)$, et $M^3=\MatricePuissancePY(1,2 § 3,4)(3)$.
```

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

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

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

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

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

LaTeX

pdfLaTeX

LuaLaTeX

TikZ

TeXLive

MiKTeX

Table des matières

Première partie
Introduction

1 Le package ResolSysteme

1.1 Introduction

La package *propose* des outils pour travailler sur des matrices ou des systèmes linéaires (de taille réduite !) :

— en calculant des produits matriciels *simples* (dimensions réduites) ;

— en affichant la **solution** (si elle existe) ;

— en affichant le **déterminant** et l'éventuelle **inverse** de la matrice des coefficients.

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

— via les packages xint* pour des formats **2x2** ou **3x3** ;

— 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

Le package 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}
```

Deuxième partie
Historique

`v0.1.3` : Ajout de commandes pour du calcul matriciel (de taille raisonnable)
`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

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

2.1 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}
```

2.2 Utilisation

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

- la version *étoilée* force l'écriture du signe « – » avant l'éventuelle fraction ;
- 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 (fraction avec - sur numérateur)
\ConvVersFrac*{-10+1/3*(-5/16)}         %sortie avec - avant la fraction
\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

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

2.3 Interaction avec les commandes « matricielles », limitations

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

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 !

3 Affichage d'une matrice

3.1 La commande

La première 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 caclulant 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)
```

3.2 Utilisation

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

- la version *étoilée* force l'écriture du signe « **−** » avant l'éventuelle fraction ;
- 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 héritée 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 $M = \begin{pmatrix} \dfrac{5}{4} & 0 & \dfrac{19}{5} \\ 0 & 1 & -\dfrac{5}{3} \\ \dfrac{1}{2} & \dfrac{9}{20} & \dfrac{6}{7} \end{pmatrix}$.

4 Calculs matriciels « simples »

4.1 Introduction

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

— des produits matriciels :
- $(1\times 2)\times(2\times 1)$;
- $(1\times 2)\times(2\times 2)$;
- $(2\times 2)\times(2\times 2)$;
- $(2\times 2)\times(2\times 1)$;
- $(1\times 3)\times(3\times 1)$;
- $(1\times 3)\times(3\times 3)$;
- $(3\times 3)\times(3\times 3)$;
- $(3\times 3)\times(3\times 1)$;

— le carré d'une matrice 2x2 ou 3x3 ;

— 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]
\CarreMatrice(*)[option de formatage]<options nicematrix>(matrice)(-5,6 § 1,4)[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, ou ne rentre pas dans le cadre des cas possibles, rien ne sera affiché !

4.2 Utilisation

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

— la version *étoilée* force l'écriture du signe « $-$ » avant l'éventuelle fraction ;

— 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 héritée 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 $\ProduitMatrices(-5,6 § 1,4)(2 § 7)$
```

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

Code LaTeX

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

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

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

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

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

5 Calcul de déterminant

5.1 Introduction

La deuxième commande (matricielle) disponible est pour calculer le déterminant d'une matrice :

— **2x2** ou **3x3** pour le package *classique* ;

— **2x2** ou **3x3** ou **4x4** pour le package *lua*.

Code LaTeX

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

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

5.2 Utilisation

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

— la version *étoilée* force l'écriture du signe « $-$ » avant l'éventuelle fraction ;

— 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 a11,a12,... § a21,a22,... (syntaxe *héritée* 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 $\mathrm{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ est $\det(\mathrm{A}) = -2$.

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

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 $\det(A) \approx 16{,}333$.

Code LaTeX

```
%version python
Le dé. de $A=\AffMatrice(1,2 § 3,4)$ est
$\det(A)=\DetMatricePY(1,2 § 3,4)$.
```

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

Code LaTeX

```
Le dét. de $A=\AffMatrice[dec](-1,0.5 § 1/2,4)$ est
$\det(A)=\DetMatricePY*[d](-1,0.5 § 1/2,4)$.
```

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

Code LaTeX

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

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

6 Inverse d'une matrice

6.1 Introduction

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

— **2x2** ou **3x3** pour le package *classique* ;

— **2x2** ou **3x3** ou **4x4** pour le package *lua*.

Code LaTeX

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

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

6.2 Utilisation

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

— la version *étoilée* force l'écriture du signe « – » avant l'éventuelle fraction ;

— 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 héritée 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 $A=\AffMatrice(1,2 § 3,4)$ est
$A^{-1}=\MatriceInverse<cell-space-limits=2pt>(1,2 § 3,4)$.
```

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

Code LaTeX

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

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

Code LaTeX

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

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

Code LaTeX

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

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 $A^{-1}=\begin{pmatrix}-15/8&-1/8&3/2&-1\\23/8&1/8&1/2&2\\-9/8&1/8&-3/2&-1\\1/8&-1/8&1/2&0\end{pmatrix}$.

7 Résolution d'un système linéaire

7.1 Introduction

La deuxième commande (matricielle) disponible est pour déterminer l'éventuelle solution d'un système linéaire qui s'écrit matriciellement A × X = B :

- **2x2** ou **3x3** pour le package *classique* ;
- **2x2** ou **3x3** ou **4x4** pour le package *lua*.

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

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

7.2 Utilisation

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

- la version *étoilée* force l'écriture du signe « **–** » avant l'éventuelle fraction ;
- 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 a11,a12,... § a21,a22,... (syntaxe héritée de `sympy`) ;
- le quatrième argument, *obligatoire* et entre (...), est quant à lui, la matrice B donnée par ses coefficients b11,b21,... (syntaxe héritée 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.

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

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

Code LaTeX

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

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

Code LaTeX

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

La solution de $\begin{cases} x + 2y + 3z + 4t = -10 \\ 5x + 6y + 7z = 0 \\ x + y + z + t = 4 \\ -2x - 3y - 5z - 6t = 7 \end{cases}$ est $\mathscr{S} = \left\{\begin{pmatrix} 17{,}75 \\ -12{,}75 \\ -1{,}75 \\ 0{,}75 \end{pmatrix}\right\}$.

Code LaTeX

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

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

Quatrième 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 lua activée, grâce notamment à la macro \py.

Code Python

```
#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 martrice
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

```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
```