

Le package `tnslinalg` : un peu d'algèbre linéaire de base

Code source disponible sur <https://github.com/typensee-latex/tnslinalg.git>.

Version 0.0.0-beta développée et testée sur Mac OS X.

Christophe BAL

2020-07-10

Table des matières

I.	Introduction	2
II.	Beta-dépendance	2
III.	Matrices via <code>nicematrix</code>	2
1.	Quelques exemples pour bien démarrer	2
2.	Calculs expliqués des déterminants 2×2	4
IV.	Historique	5
V.	Toutes les fiches techniques	6
1.	Matrices via <code>nicematrix</code>	6
i.	Calculs expliqués des déterminants 2×2	6

I. Introduction

Le package `tnslinalg` complète un tout petit peu l'excellent package `nicematrix` qui est importé par `tnscom` (*voir la section suivante à ce sujet*).

II. Beta-dépendance

`tnscom` qui est disponible sur <https://github.com/typensee-latex/tnscom.git> est un package utilisé en coulisse.

III. Matrices via `nicematrix`

Le gros du boulot est fait par l'excellent package `nicematrix`¹. `tnslinalg` propose en plus une macro à but pédagogique : voir la section 2. page 4. Veuillez vous reporter à la documentation de `nicematrix` pour savoir comment s'y prendre en général.

1. Quelques exemples pour bien démarrer

Exemple 1 – Vu dans la documentation de `nicematrix`

```
\begin{pmatrix}
  1      & \cdots & \cdots & 1      & \\
  0      & \ddots &      & \vdots & \\
  \vdots & \ddots & \ddots & \vdots & \\
  0      & \cdots & 0      & 1      & \\
\end{pmatrix}
```

$$\begin{pmatrix} 1 & \cdots & \cdots & 1 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

Exemple 2

```
\begin{vmatrix}
  1      & \cdots & \cdots & 1      & \\
  0      & \ddots &      & \vdots & \\
  \vdots & \ddots & \ddots & \vdots & \\
  0      & \cdots & 0      & 1      & \\
\end{vmatrix}
```

$$\begin{vmatrix} 1 & \cdots & \cdots & 1 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{vmatrix}$$

Exemple 3

```
\begin{bmatrix}
  1      & \cdots & \cdots & 1      & \\
  0      & \ddots &      & \vdots & \\
  \vdots & \ddots & \ddots & \vdots & \\
  0      & \cdots & 0      & 1      & \\
\end{bmatrix}
```

$$\begin{bmatrix} 1 & \cdots & \cdots & 1 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

1. On impose l'option `transparent`.

Exemple 4 – Vu dans la documentation de nicematrix

```

 $\begin{pNiceMatrix}[name = mymatrix]$ 
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9
 $\end{pNiceMatrix}$ 

\tikz[remember picture,
  overlay]
\draw[red]
  (mymatrix-2-2) circle (2.5mm);

```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & \textcircled{5} & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Exemple 5 – Vu dans la documentation de nicematrix

```

 $\left($ 
  \begin{NiceArray}{cccc:c}
    1 & 2 & 3 & 4 & 5 \\
    6 & 7 & 8 & 9 & 10 \\
    11 & 12 & 13 & 14 & 15
  \end{NiceArray}
 $\right)$ 

```

$$\left(\begin{array}{cccc:c} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{array} \right)$$

Exemple 6 – Proposition de l’auteur de nicematrix suite à une discussion par mail

```

% Besoin du package ‘‘ifthen‘‘.
\newcommand\aij{%
  a_{\arabic{iRow}\arabic{jCol}}%
}

 $\begin{bNiceArray}{*{5}{>{%$ 
  \ifthenelse{\value{iRow}>0}{\aij}{}%
}{c}}[
  first-col,
  first-row,
  code-for-first-row
    = \mathbf{\arabic{jCol}},
  code-for-first-col
    = \mathbf{\arabic{iRow}}
]

& & & & \\
& & & & \\
& & & &
\end{bNiceArray}

```

$$\begin{matrix} & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} \\ \mathbf{1} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ \mathbf{2} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \end{matrix}$$

Exemple 7 – Avec des calculs automatiques

```
\newcounter{cntaij}
\newcommand\aij{%
  \setcounter{cntaij}{\value{iRow}}%
  \addtocounter{cntaij}{\value{jCol}}%
  \addtocounter{cntaij}{-1}%
  \arabic{cntaij}%
}

Si $a_{ij} = i + j - 1$ alors

$(a_{ij})_{1 \leq i \leq 3, 1 \leq j \leq 5}$

=

\begin{bNiceArray}{*{5}{>\aij}c}}
& & & & \\
& & & & \\
& & & & 
\end{bNiceArray}$
```

Si $a_{ij} = i + j - 1$ alors

$$(a_{ij})_{1 \leq i \leq 3, 1 \leq j \leq 5} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \end{bmatrix}$$

2. Calculs expliqués des déterminants 2×2

Exemple

```
$\calcdettwo* \quad {a}{c}%
\quad {b}{d}
= \calcdettwo \quad {a}{c}%
\quad {b}{d}
= \calcdettwo[exp]{a}{c}%
\quad {b}{d}$
```

$$\begin{vmatrix} a & c \\ b & d \end{vmatrix} = \begin{vmatrix} a & c \\ b & -d \end{vmatrix} = a d - b c$$

Remarque. Il existe deux autres types de développement.

1. $a \cdot d - b \cdot c$ s'obtient via l'option `cexp`.
2. $a \times d - b \times c$ s'obtient via l'option `texp`

`exp` est pour `exp-and` soit « développer » en anglais, `c` pour `\cdot` et enfin `t` pour `\times`.

IV. Historique

Nous ne donnons ici qu'un très bref historique récent² de `tnslinalg` à destination de l'utilisateur principalement. Tous les changements sont disponibles uniquement en anglais dans le dossier `change-log` : voir le code source de `tnslinalg` sur [github](#).

2020-07-10 Première version 0.0.0-beta.

2. On ne va pas au-delà de un an depuis la dernière version.

V. Toutes les fiches techniques

1. Matrices via `nicematrix`

i. Calculs expliqués des déterminants 2×2

`\calcdettwo` `[#opt]` `{#1..#4}`

`c` = `c-alculate`

`\calcdettwo*` `[#opt]` `{#1..#4}`

— **Option**: la valeur par défaut est `std` pour `standard`. Voici les différentes valeurs possibles.

1. `std` : on utilise l'écriture matricielle.
2. `exp` : ceci demande d'afficher une formule développée en utilisant \times pour les produits.
3. `cexp` : comme `exp` mais avec le symbole \cdot obtenu via `\cdot`.
4. `semp` : comme `exp` mais avec un espace pour séparer les facteurs de chaque produit.

— **Argument 1**: l'entrée à la position $(1, 1)$

— **Argument 2**: l'entrée à la position $(1, 2)$

— **Argument 3**: l'entrée à la position $(2, 1)$

— **Argument 4**: l'entrée à la position $(2, 2)$