

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

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

Version 0.1.0-beta développée et testée sur Mac OS X.

Christophe BAL

2020-08-27

---

## Table des matières

I. Introduction	2
II. Beta-dépendance	2
III. Packages utilisés	2
IV. Matrices via <code>nicematrix</code>	2
1. Quelques exemples pour bien démarrer	2
2. Calcul expliqué d'un déterminant $2 \times 2$	5
V. Historique	6
VI. Toutes les fiches techniques	7
1. Matrices via <code>nicematrix</code>	7
i. Calcul expliqué d'un déterminant $2 \times 2$	7

---

# 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. Packages utilisés

La roue ayant déjà été inventée, le package `tnslinalg` réutilise les packages suivants sans aucun scrupule.

- `commado`
- `etoolbox`
- `xstring`

## IV. Matrices via `nicematrix`

Le gros du boulot est fait par l'excellent package `nicematrix`<sup>1</sup>. `tnslinalg` propose en plus une macro à but pédagogique : voir la section 2. page 5. 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`**

<pre><code>\$\begin{pmatrix}</code> 1 &amp; <code>\cdots</code> &amp; <code>\cdots</code> &amp; 1 &amp; <code>\\\</code> 0 &amp; <code>\ddots</code> &amp; &amp; <code>\vdots</code> &amp; <code>\\\</code> <code>\vdots</code> &amp; <code>\ddots</code> &amp; <code>\ddots</code> &amp; <code>\vdots</code> &amp; <code>\\\</code> 0 &amp; <code>\cdots</code> &amp; 0 &amp; &amp; 1 <code>\end{pmatrix}</code>\$</pre>	$\begin{pmatrix} 1 & \cdots & \cdots & 1 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{pmatrix}$
---	---

**Exemple 2**

<pre><code>\$\begin{vmatrix}</code> 1 &amp; <code>\cdots</code> &amp; <code>\cdots</code> &amp; 1 &amp; <code>\\\</code> 0 &amp; <code>\ddots</code> &amp; &amp; <code>\vdots</code> &amp; <code>\\\</code> <code>\vdots</code> &amp; <code>\ddots</code> &amp; <code>\ddots</code> &amp; <code>\vdots</code> &amp; <code>\\\</code> 0 &amp; <code>\cdots</code> &amp; 0 &amp; &amp; 1 <code>\end{vmatrix}</code>\$</pre>	$\begin{vmatrix} 1 & \cdots & \cdots & 1 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{vmatrix}$
---	---

---

1. On impose l'option `transparent`.

### Exemple 3

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

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

### 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);
```

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

### 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)
```

$$\begin{pmatrix} 1 & 2 & 3 & 4 & : & 5 \\ 6 & 7 & 8 & 9 & : & 10 \\ 11 & 12 & 13 & 14 & : & 15 \end{pmatrix}$$

## 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}}
]

& & & & \backslash\backslash
& & & & \backslash\backslash
& & & &
\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}}
& & & & \backslash\backslash
& & & & \backslash\backslash
& & & &
\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. Calcul expliqué d'un déterminant $2 \times 2$

### Exemple 1 – Versions matricielles

<pre> \$\calcdettwo{a}{b}%     {c}{d}\$  ou  \$\calcdettwo[loop]{a}{b}%     {c}{d}\$  ou  \$\calcdettwo[arrows]{a}{b}%     {c}{d}\$  ou  \$\calcdettwo[cross]{a}{b}%     {c}{d}\$ </pre>	
--	--

### Exemple 2 – Versions développées

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

<pre> \$\calcdettwo[exp]{a}{b}%     {c}{d}\$  ou  \$\calcdettwo[texp]{a}{b}%     {c}{d}\$  ou  \$\calcdettwo[cexp]{a}{b}%     {c}{d}\$ </pre>	$ad - cb$ ou $a \times d - c \times b$ ou $a \cdot d - c \cdot b$
---	---

## V. Historique

Nous ne donnons ici qu'un très bref historique récent<sup>2</sup> 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](#).

**2020-08-27** Nouvelle version mineure `0.1.0-beta`.

- **DÉTERMINANT  $2 \times 2$**  : changement de l'API.
    - `\calcdettwo` sert à obtenir au choix les versions développée ou bien celles matricielles avec pour décorations supplémentaires une croix fléchée ou non.
    - Suppression de `\calcdettwo*`.
- 

**2020-07-10** Première version `0.0.0-beta`.

---

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

## VI. Toutes les fiches techniques

### 1. Matrices via `nicematrix`

#### i. Calcul expliqué d'un déterminant $2 \times 2$

`\calcdettwo` [`#opt`] {`#1..#4`} `c = c-alculate`

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

1. `arrows` : des croix fléchées indiquent comment effectuer les calculs.
2. `cross` : des croix non fléchées indiquent comment effectuer les calculs.
3. `loop` : des boucles indiquent comment effectuer les calculs.
4. `nodeco` : rien n'indique comment effectuer les calculs.
5. `exp` : ceci demande d'afficher une formule développée en utilisant un espace pour séparer les facteurs de chaque produit.
6. `cexp` : comme `exp` mais avec le symbole  $\cdot$  obtenu via `\cdot`.
7. `texp` : comme `exp` mais avec le symbole  $\times$ .

— **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)