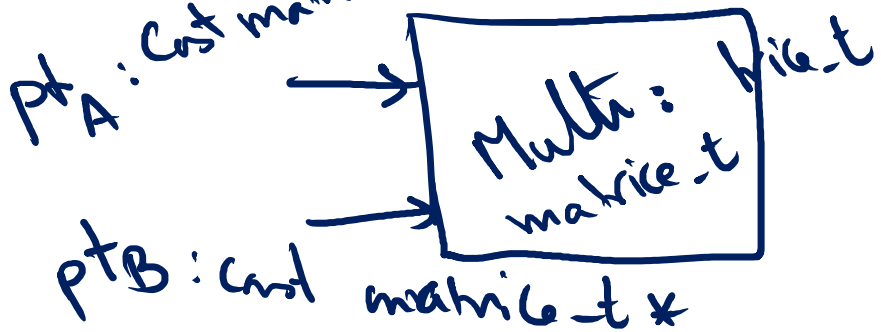
1. Multiplication de deux matrices

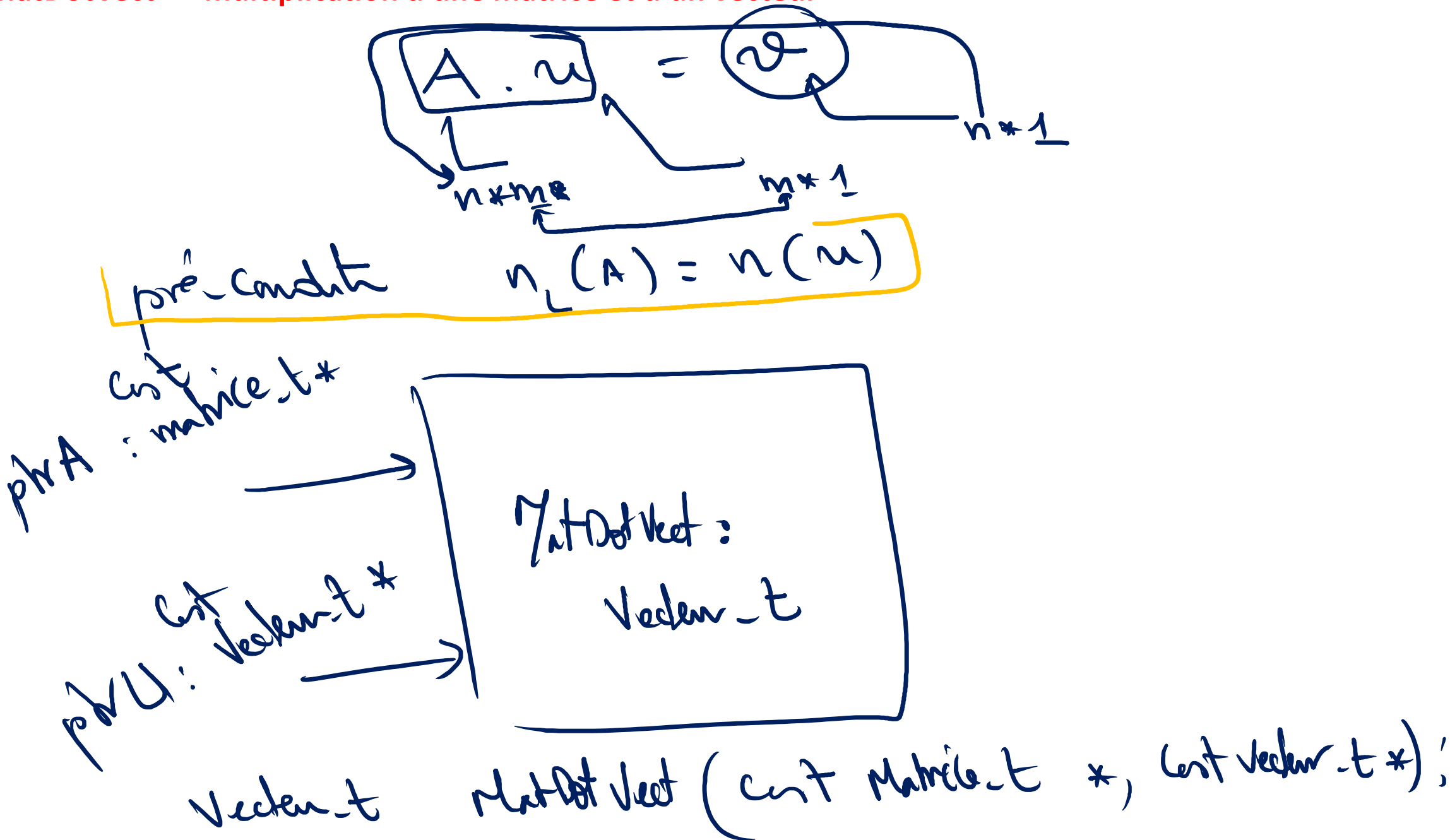
$C = A\,B$ $\quad A_{n*m}$, B_{m*l} , C_{n*l}

pré_Condition : $m_C(A) = n_L(B)$




```
matrice_t multiplicat (cst matrice_t *, cst matrice_t *);
```

2. MatDotVect => multiplication d'une matrice et d'un vecteur

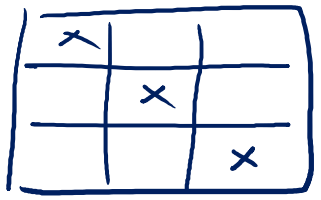


3. vecteur getDiag => extraction de la 1ère diagonale d'une matrice carrée

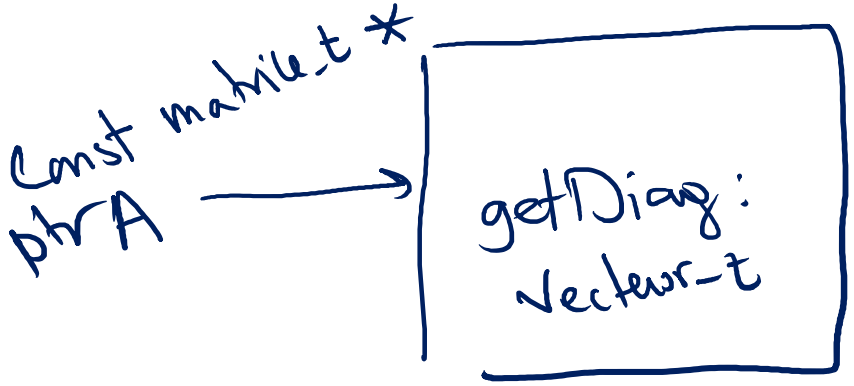
V = getDiag (A) ⇒ Vecteur

Vecteur n×1

matrice carrée n×n



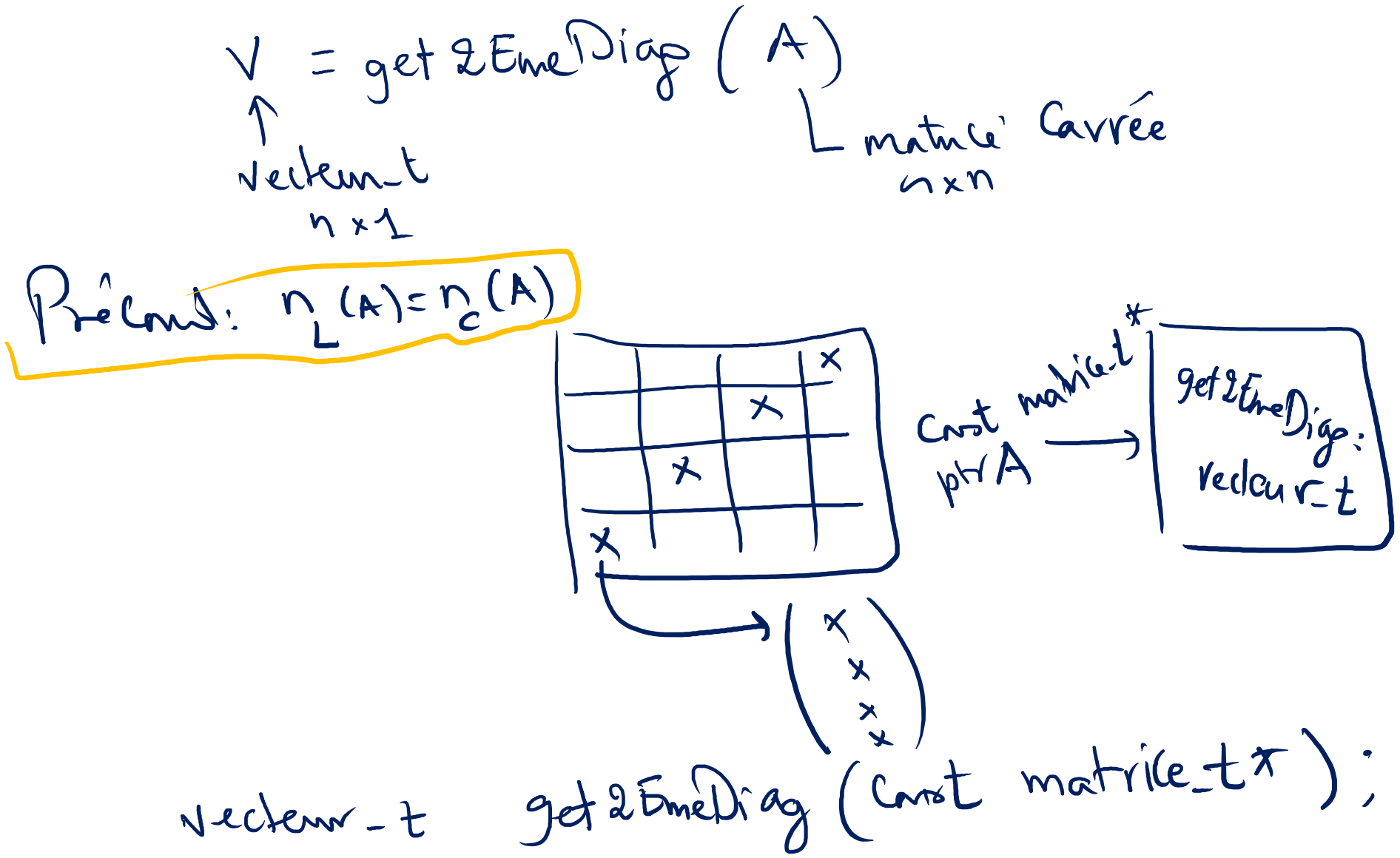
⇒ $\begin{pmatrix} x \\ x \\ x \end{pmatrix}$



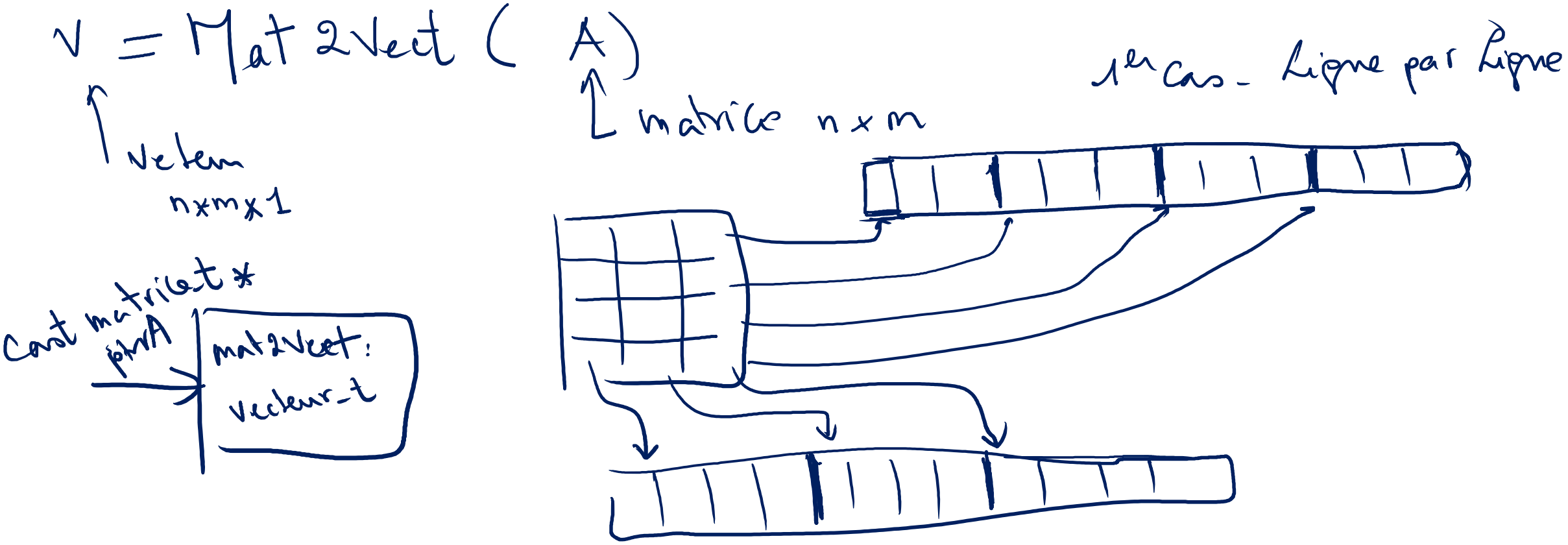
Précond : $n_L(A) = n_C(A)$

Vecteur_t getDiag (const matrice_t&) ;

4. vecteur get2emeDiag => Extraire la 2ème diagonale d'une matrice carrée



5. Mat2Vect => Transformer une matrice en un vecteur (en justaposant les lignes ou les colonnes)



6. Vect2Mat => Transformer un vecteur en une matrice(verifier la possibilité de la transforamation nl*nc=nbElts)

entier

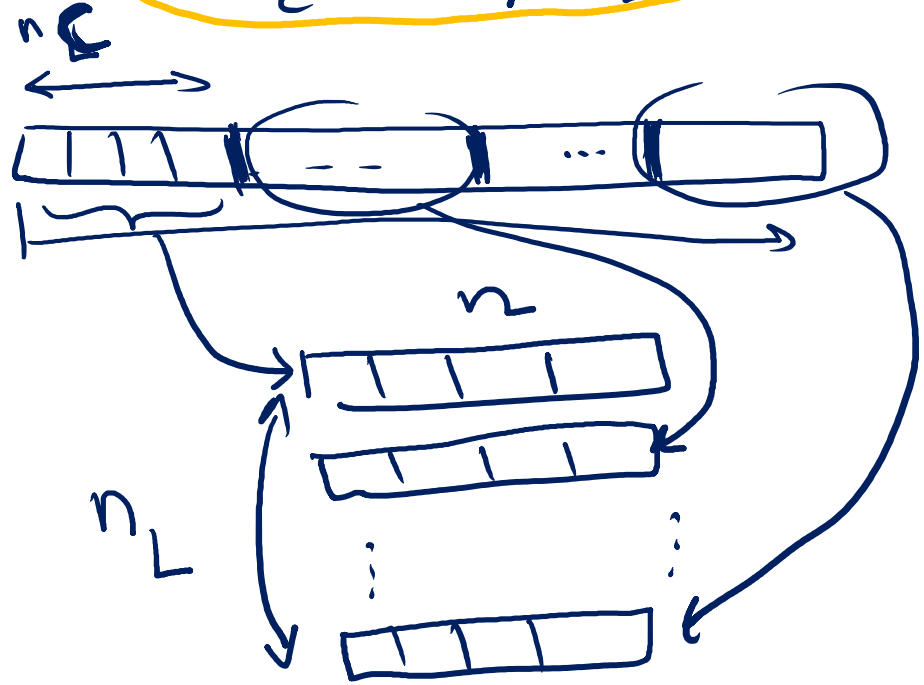
A = vect2Mat (V , nL)

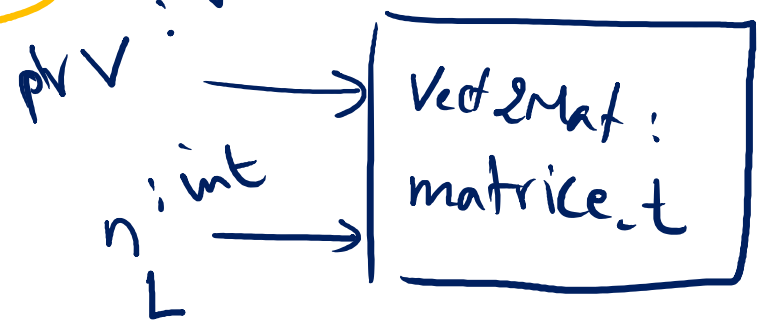
A : matrice ($n_L \times n_c$)

V : Vecteur ($n \times 1$)

Précond : n_L diviseur de n

$n_c = n / n_L$





matrice_t vect2Mat(const vecteur_t *
, int);

7. getTrace => calculer la somme des éléments de la diagonale d'une matrice carrée

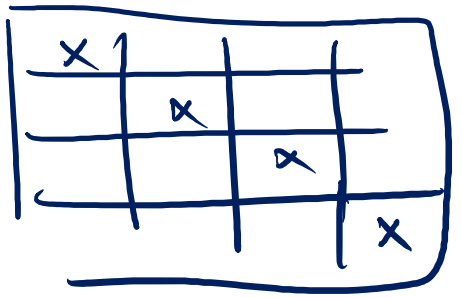
$tr = getTrace(A) = \sum getDiag(A)$

tr : Réel

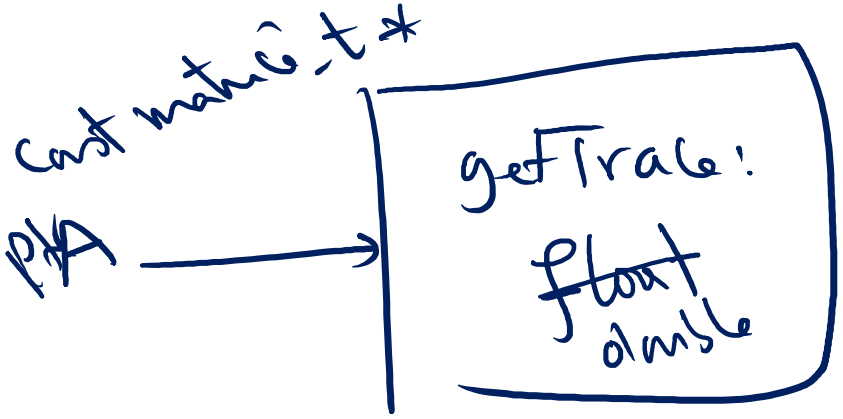
A : matrice $n \times n$

Précond $n_c(A) == n_L(A)$

A



$\Rightarrow \sum \begin{pmatrix} x \\ x \\ x \\ x \end{pmatrix}$



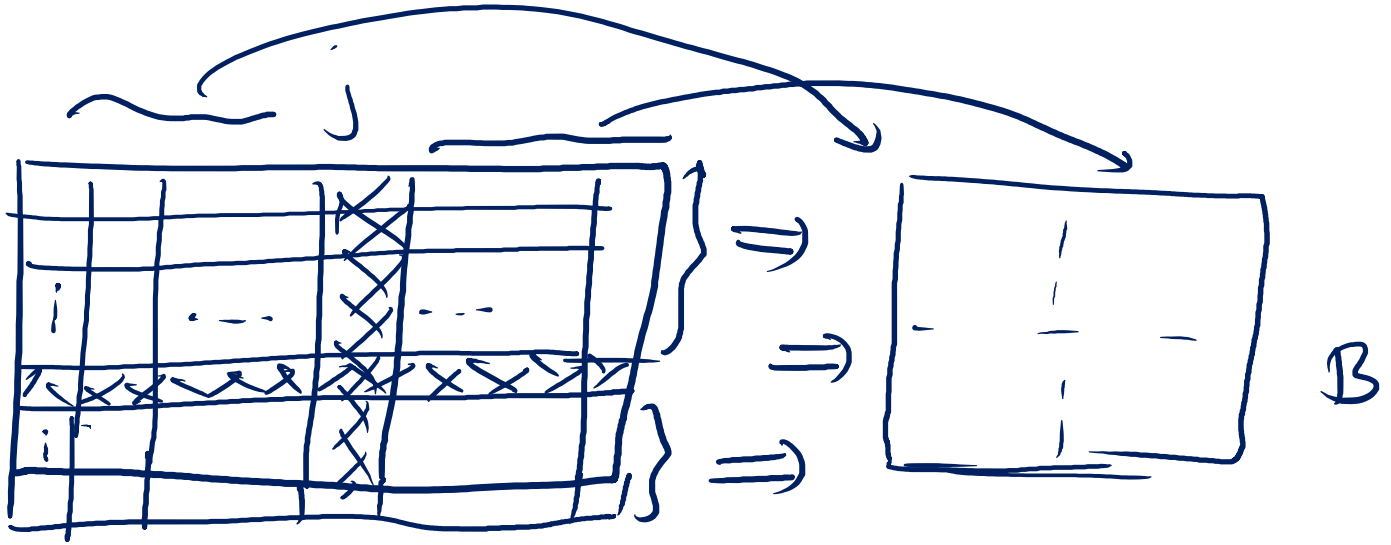
double getTrace (const matrice_t*);

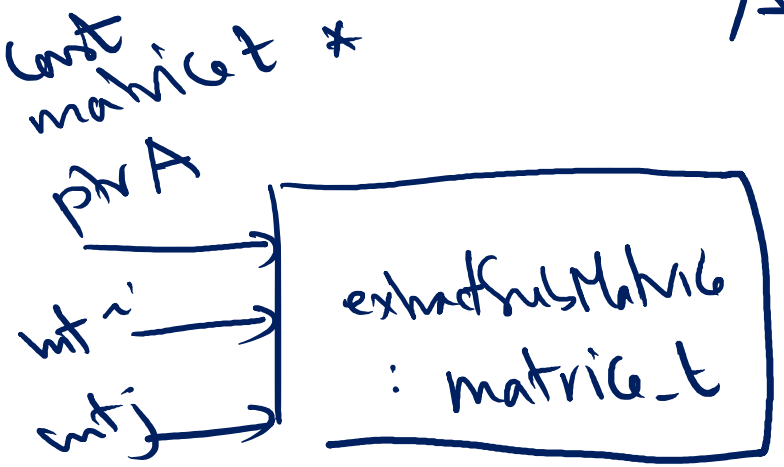
8. extractSubMatrice => extraire la sous matrice d'une matrice en eliminant la ligne I et la colonne j

B = extractSubMatrice(A, i, j)

matrice (n-1) x (m-1)

matrice nxm



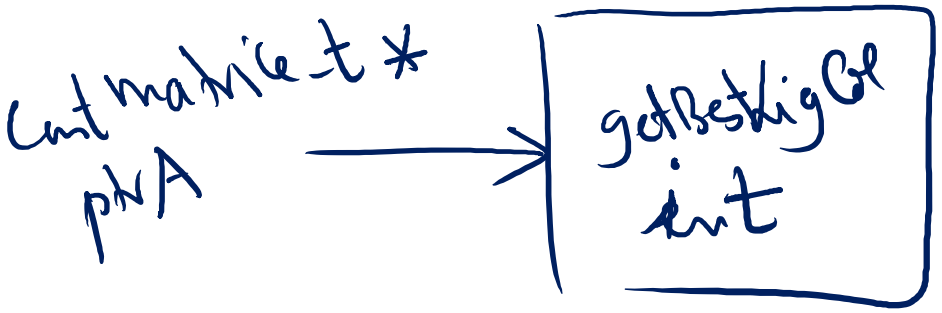


matrice_t extractSubMatrice(const matrice_t *, int, int);

9. getBestligneCol => trouver la ligne ou la colonne d'une matrice qui contient le maximum de zéros

numero = getBestLigneCol (A)
↑ matrice n x m
entier ∈ ℤ

. on convient de retourner un entier négatif pour les lignes et positif pour les colonnes.



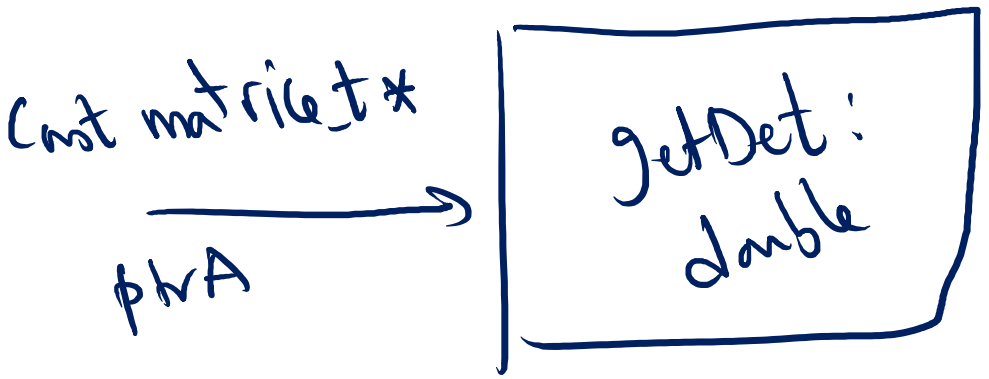

```
int getBestLigCol(const matrice_t *);
```

10. getDet => calculer le determinant d'une matrice carrée

det = getDet(A) — Matrice ($n \times n$)

↑ Réel

Précond : $n_L(A) = n_C(A)$



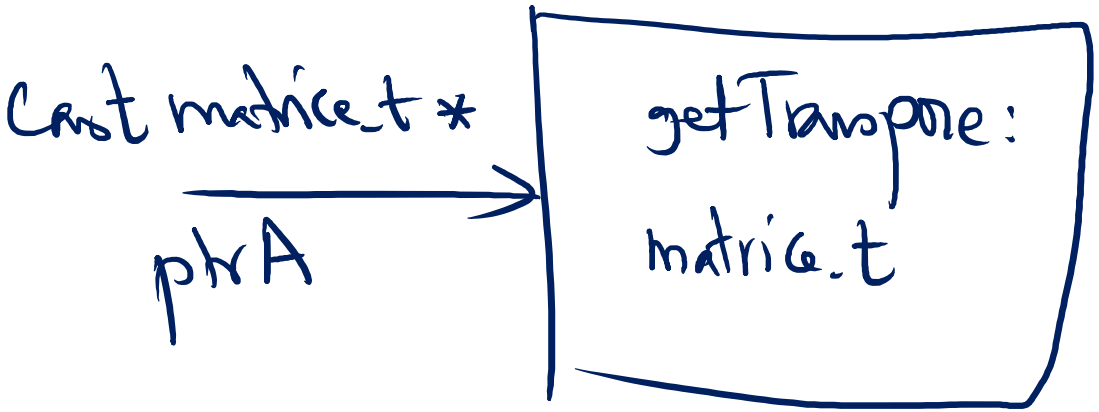

```
double getDet(Const matrice_t *);
```

11. getTranspose => calculer la matrice transpose d'une matrice

$$B = A^T = \text{getTranspose}(A)$$

B : matrice $(m \times n)$

A : Matrice $(n \times m)$




```
matrice_t getTranspose(cast matrice_t *);
```

12. getInverse => Calculer la matrice inverse d'une matrice carrée si le determinant est non null

$B = A^{-1} = \text{getInverse}(A)$

- B : Matrice carrée $n \times n$
- A : matrice carrée $n \times n$

Preconditions :

$n_L(A) = n_C(A)$

$\det(A) != 0$

Cast matrice_t * ptrA → getInverse : matrice_t

```
matrice_t getInverse(const matrice_t *);
```

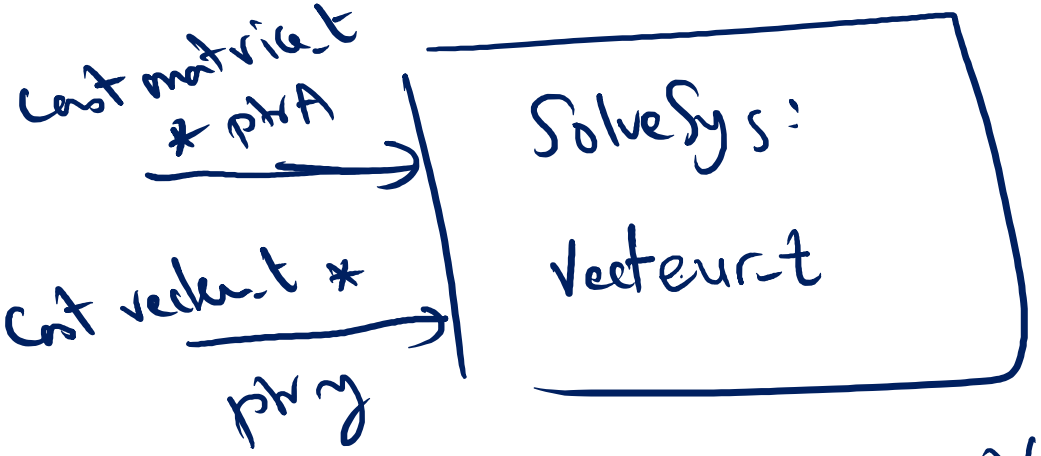
13. SolveSys => résoudre un système linéaire $A_{nxn}X_{nx1}=Y_{nx1}$

$$A\,x = y \Rightarrow x = \text{SolveSys}(A, y)$$

$$\Downarrow$$

$$x = A^{-1}\,y \Rightarrow x = \text{MatDotVect}(\text{getInverse}(A), y)$$

Préconds = Precond des fonctions MatDotvet et getInverse



Vecteur_t SolveSys(Const matrice_t *, Const vecteur_t *);