

Matrice ! Matrice ! est-ce que j'ai une gueule de Matrice ?

Commençons par une petite pensée philosophique :

“Les mathématiques peuvent être définies comme une science dans laquelle on ne sait jamais de quoi on parle, ni si ce qu'on dit est vrai”.

(Bertrand Russel (1872-1970) Artiste, écrivain, Homme politique, Mathématicien, Moraliste, Philosophe, Scientifique)

O Préambule	P 2
I Pourquoi s'intéresser à ces structures barbares ?	P 3
1.1 Quelques exemples introductifs.	P 3
1.1.1 Les systèmes d'équations linéaires	P 3
1.1.2 Les changements de coordonnées	P 4
1.2 Mais en fait, c'est quoi une matrice ?	P 4
II Les opérations de base sur les matrices	P 5
2.1 Somme de matrices	P 5
2.1.1 Exemple	P 5
2.1.2 Généralisation	P 6
2.2 Multiplication	P 6
2.2.1 Exemple	P 6
2.2.2 Généralisation	P 9
2.3 Multiplication par un nombre (élément)	P 10
2.3.1 Exemple	P 10
2.3.2 Généralisation	P 10
2.4 Déterminant d'une matrice	P 11
2.4.1 Exemple	P 11
2.4.2 Généralisation	P 11
2.4.3 Propriétés du déterminant :	P 12
2.4.4 Efficacité du calcul du déterminant	P 12
III Matrices particulières	P 14
3.1 Matrice unité	P 14
3.2 Transposée d'une matrice	P 14
3.3 L'inverse d'une matrice	P 14
3.4 Matrice trigonale supérieure (U) up	P 15
3.5 Matrice trigonale inférieure (L) low	P 15
3.6 Matrice diagonale	P 15
IV Résolution d'un système linéaire	P 16
4.1 Exemple introductif	P 17
4.2 Solution mathématique brute	P 17
4.3 Méthode de Kramer	P 17
4.3.1 Effectuons le calcul sur un exemple	P 18
4.3.2 Généralisation	P 18
4.3.3 Algorithmique.	P 18
4.4 Elimination de gauss-Jordan par pivotement partiel	P 18
4.4.1 Effectuons le calcul sur un exemple	P 19
4.4.2 Généralisation et algorithme	P 21

4.4.2.1 La trigonalisation	P 21
4.4.2.2 Le calcul inverse des résultats	P 22
4.4.3 Efficacité de la méthode	P 22
4.5 Décomposition L(lower)U(upper) par méthode de Crout	P 23
4.5.1 Exemple introductif	P 24
4.5.2.1 En restant simple	P 24
4.5.2.2 Pour les adeptes de la migraine	P 25
4.6 Comparaison des méthodes	P 26
V Utilisation des matrices en géométrie vectorielle	P 27
5.1 Exemple introductif	P 27
5.2 Rotation autour d'un axe	P 27
5.2.1 Axe \vec{z}	P 27
5.2.2 Axe \vec{x}	P 28
5.2.3 Axe \vec{y}	P 28
5.2.4 Axe quelconque \vec{u}	P 28
5.3 Symétries planes	P 29
5.3.1 Par rapport au plan \vec{x}, \vec{y} de normale \vec{z}	P 30
5.3.2 Par rapport au plan \vec{y}, \vec{z} de normale \vec{x}	P 30
5.3.3 Par rapport au plan \vec{z}, \vec{x} de normale \vec{y}	P 30
5.3.4 Par rapport à un plan de normale \vec{n}	P 30
5.4 Homothéties	P 30
5.4.1 Globale	P 31
5.4.2 Dans une direction \vec{n}	P 31
5.4.3 Projection sur un plan de normale \vec{n}	P 31
5.5 Qu'en est-il des opérations inverses ?	P 31
VI A propos de matrices homogènes	P 32
6.1 Exemple introductif!!	P 32
6.2 Translation homogène	P 33
6.3 Autres manipulations homogènes	P 33
6.4 Domaines d'utilisation du système « homogène »	P 33
6.5 soyons Fou	P 34
VII Les changements de repères	P 34
7.1 Exemple introductif	P 35
7.2 Généralisation	P 36
Note du lecteur	P 38
Note de l'auteur	P 38

O Préambule

Encore un cours sur les matrices me direz-vous ! Oui et non, dans les pages qui vont suivre (qui au départ n'étaient vouées qu'à être quelques lignes de clarification !), je vais essayer d'aborder le concept et les outils afférant aux matrices. Nous essayerons de partir de l'exemple et ensuite de le généraliser sans pour autant insister (ce qui va faire hurler les puristes mathématiciens, mais j'assume !) sur des démonstrations fastidieuses et inutiles (à mon sens) lorsqu'il s'agit d'utiliser l'outil. En aucun cas vous n'entendrez les mots d'espace vectoriel et consorts. Cet article a été écrit dans l'esprit d'une présentation d'un ingénieur pour des ingénieurs. Il peut néanmoins être lu en partie (notamment les parties exemples) par un néophyte qui comprendrait le concept par l'exemple. Au niveau notations, une part importante des descriptions fait appel aux notations avec indices. Pour les

néophytes, ne vous affolez pas, les exemples permettent de se familiariser progressivement avec ces notations.

I Pourquoi s'intéresser à ces structures barbares ?

1.1 Quelques exemples introductifs.

1.1.1 Les systèmes d'équations linéaires :

Le système d'équations $\begin{cases} 3x - 2y = 2 \\ z - y = 3 \\ x - 2z = 4 \end{cases}$ donne comme solution par substitutions et beaucoup d'erreurs de

calculs manuels possibles : $\begin{cases} x = -4 \\ y = -7 \\ z = -4 \end{cases}$

Réorganisons un peu ces équations $\begin{cases} 3x - 2y & & = 2 \\ & -y & +z = 3 \\ x & & -2z = 4 \end{cases}$

Ou encore un peu plus en systématisant $\begin{cases} 3x - 2y + 0z = 2 \\ 0x - y + z = 3 \\ x + 0y - 2z = 4 \end{cases}$

Ce système peut s'écrire sous la forme synthétique $\begin{bmatrix} 3 & -2 & 0 \\ 0 & -1 & 1 \\ 1 & 0 & -2 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$ avec $M = \begin{bmatrix} 3 & -2 & 0 \\ 0 & -1 & 1 \\ 1 & 0 & -2 \end{bmatrix}$ le tableau

des coefficients des équations ordonnées, $X = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$ le tableau vertical des inconnues, $C = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$ le tableau vertical

des constantes du deuxième membre.

$$M \times X = C$$

Avec la règle de multiplication de ces tableaux $M \times X = \begin{bmatrix} 3 & -2 & 0 \\ 0 & -1 & 1 \\ 1 & 0 & -2 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 3 \cdot x - 2 \cdot y + 0 \cdot z \\ 0 \cdot x - 1 \cdot y + 1 \cdot z \\ 1 \cdot x + 0 \cdot y - 2 \cdot z \end{bmatrix}$ (on parle de

produit ligne par colonne).

L'égalité entre deux colonnes se faisant terme à terme $M \times X = C$ est équivalent à

$$\begin{bmatrix} 3 \cdot x - 2 \cdot y + 0 \cdot z \\ 0 \cdot x - 1 \cdot y + 1 \cdot z \\ 1 \cdot x + 0 \cdot y - 2 \cdot z \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} \Leftrightarrow \begin{cases} 3 \cdot x - 2 \cdot y + 0 \cdot z = 2 \\ 0 \cdot x - 1 \cdot y + 1 \cdot z = 3 \\ 1 \cdot x + 0 \cdot y - 2 \cdot z = 4 \end{cases}$$

qui, une fois ordonné et bien écrit, est notre système d'équations initial.

M est une collection bidimensionnelle d'objets numérique appelée matrice carrée (autant de lignes que de colonnes).

X est une collection unidimensionnelle d'objets inconnus (qui devraient être des nombres en fin de parcours) appelé matrice colonne ou vecteur.

C est une collection unidimensionnelle d'objets numériques : vecteur constant.

La résolution de ce système revient à résoudre l'équation $M \times X = C$ et donc en supposant que M possède une inverse au sens de la multiplication des matrices (nous y reviendrons !!), $M^{-1} \times M = M \times M^{-1} = I$ (la matrice identité au sens de la multiplication des matrices), le système se résout par $M^{-1} \times M \times X = M^{-1} \times C$ soit

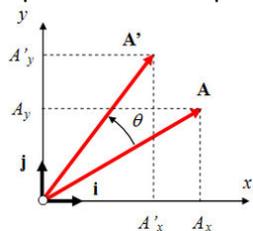
$$I \times X = M^{-1} \times C \text{ ou finalement } X = M^{-1} \times C$$

Reste à déterminer M^{-1} en fonction de M .

En pratique, il n'y a qu'en cours de mathématique que l'on traite des systèmes à trois inconnues ! Dans la vraie vie d'ingénierie, soit le système est simpliste (une équation une inconnue) ou ultra complexe (1000 à 10000 équations et donc 1000 à 10000 inconnues). Dans le premier cas toute débauche de théorie sur les matrices est inutile. Dans le deuxième cas, seul le traitement informatique, et donc une formalisation du problème et une généralisation donnant lieu à l'application numérique du développement théorique des propriétés des matrices est envisageable.

1.1.2 Les changements de coordonnées :

Repérer un point dans l'espace à l'aide de valeurs numériques est désormais une habitude de tous les instants lorsqu'on fait de la géo-localisation ou lorsqu'on cherche une trajectoire d'un objet, d'un bras robot ou que autre. Le travail des repères est fastidieux mais peut être facilité par l'utilisation de l'outil matriciel qui permet de représenter un déplacement.



On désire connaître les coordonnées du point A' en fonction de celles de A après rotation d'un angle θ .

$$\overrightarrow{OA} = \begin{pmatrix} A_x \\ A_y \end{pmatrix}_{\vec{x}, \vec{y}} = A_x \cdot \vec{x} + A_y \cdot \vec{y} \text{ la rotation de } \theta \text{ des vecteurs unitaires } \vec{x} \text{ et } \vec{y} \text{ donne les vecteurs}$$

$$\vec{x}' \text{ et } \vec{y}' \text{ tel que } \begin{cases} \vec{x}' = \cos(\theta) \cdot \vec{x} + \sin(\theta) \cdot \vec{y} \\ \vec{y}' = -\sin(\theta) \cdot \vec{x} + \cos(\theta) \cdot \vec{y} \end{cases} \text{ et donc}$$

$$\overrightarrow{OA'} = \begin{pmatrix} A'_x \\ A'_y \end{pmatrix}_{\vec{x}', \vec{y}'} = \begin{pmatrix} A_x \\ A_y \end{pmatrix}_{\vec{x}, \vec{y}} = A_x \cdot \vec{x}' + A_y \cdot \vec{y}' = A_x \cdot (\cos(\theta) \cdot \vec{x} + \sin(\theta) \cdot \vec{y}) + A_y \cdot (-\sin(\theta) \cdot \vec{x} + \cos(\theta) \cdot \vec{y})$$

$$\overrightarrow{OA'} = \begin{pmatrix} A_x \cdot \cos(\theta) - A_y \cdot \sin(\theta) \\ A_x \cdot \sin(\theta) + A_y \cdot \cos(\theta) \end{pmatrix}_{\vec{x}, \vec{y}}$$

Cette transformation peut être aisément représentée et calculée à l'aide du produit maintenant bien connu (sic) :

$$\overrightarrow{OA'} = \begin{pmatrix} A'_x \\ A'_y \end{pmatrix}_{\vec{x}', \vec{y}'} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{pmatrix} A_x \\ A_y \end{pmatrix}_{\vec{x}, \vec{y}} \cdot \overrightarrow{OA'} = M_{rot(\theta)} \overrightarrow{OA} \text{ Cette représentation révèle toute sa force en}$$

3D (ou l'on trouve une formulation analogue) et lors de mouvements combinés, comme dans le cas des bras de robots articulés. Deux rotations successives de même centre O se traduisent par une simple multiplication de matrice $\overrightarrow{OA''} = M_{rot(\beta)} (M_{rot(\theta)} (\overrightarrow{OA}))$ décrit par la fonction matricielle $\overrightarrow{OA''} = M_{rot(\beta)} \times M_{rot(\theta)} \times \overrightarrow{OA}$ avec toute la rigueur de la multiplication des matrices.

1.2 Mais en fait c'est quoi une matrice ?

Depuis quelques lignes on vous parle sans cesse matrice. Vous l'avez sûrement compris, une matrice est un tableau d'objets, ordonné en lignes et colonnes. Pour pouvoir travailler avec ces « tableaux », les objets les constituants doivent avoir la possibilité de s'additionner et de se multiplier entre eux (ce peut être une addition et une multiplication autre que celle que nous connaissons avec les nombres, mais qui conserve des règles similaires - théorie des groupes-). Nous pouvons donc envisager des matrices de matrices de nombres ...

Une matrice sera notée par la suite dans les développements par $A_{l,c}$ $\left\{ \begin{array}{l} A : \text{le nom de "baptême"} \\ l : \text{nombre de lignes} \\ c : \text{nombre de colonnes} \end{array} \right.$.

Les éléments de cette matrice $a_{i,j}$ $\left\{ \begin{array}{l} a : \text{le nom de "baptême" de l'élément} \\ i : \text{indice de ligne} \\ j : \text{indice de colonne} \end{array} \right.$ au niveau des indices, nous prendrons

comme convention qu'ils varient de 1 à max (attention, en informatique suivant le langage utilisé, l'indice du premier élément peut être de 0 ou 1 et celui du dernier size-1 ou size lorsque size représente le nombre d'éléments du tableau). Soit les notations $A_{n,m} = [a_{i,j}]_{n,m} = [\{a_{i,j}\}]_{n,m}$. Nous utiliserons cette dernière notation dans le cas d'une matrice dont les sous éléments peuvent être des objets plus complexes qu'un nombre i.e., une collection ordonnée d'objets.

II Les opérations de base sur les matrices

2.1 Somme de matrices

2.1.1 Exemple

Faisons simple :

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}; B = \begin{bmatrix} 4 & 3 \\ 2 & 6 \end{bmatrix}$$

$$A + B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 4 & 3 \\ 2 & 6 \end{bmatrix} = \begin{bmatrix} 1+4 & 2+3 \\ 3+2 & 4+6 \end{bmatrix} = \begin{bmatrix} 5 & 5 \\ 5 & 10 \end{bmatrix}$$

$$B + A = \begin{bmatrix} 4 & 3 \\ 2 & 6 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 4+1 & 3+2 \\ 2+3 & 6+4 \end{bmatrix} = \begin{bmatrix} 5 & 5 \\ 5 & 10 \end{bmatrix} = A + B$$

Vous l'avez compris, il s'agit d'une addition classique terme à terme de structures semblables.

Plus compliqué :

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}; B = \begin{bmatrix} 4 & 3 \\ 2 & 6 \end{bmatrix}; I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} A & I \\ I & B \end{bmatrix}; D = \begin{bmatrix} I & B \\ I & B \end{bmatrix}$$

$$E = C + D = \begin{bmatrix} A+I & I+B \\ I+I & B+B \end{bmatrix}$$

$$E_{2,2,2,2,2} = \left[\begin{array}{c|c} \left[\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array} \right] + \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right] & \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right] + \left[\begin{array}{cc} 4 & 3 \\ 2 & 6 \end{array} \right] \\ \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right] + \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right] & \left[\begin{array}{cc} 4 & 3 \\ 2 & 6 \end{array} \right] + \left[\begin{array}{cc} 4 & 3 \\ 2 & 6 \end{array} \right] \end{array} \right] = \left[\begin{array}{c|c} \left[\begin{array}{cc} 2 & 2 \\ 3 & 5 \end{array} \right] & \left[\begin{array}{cc} 5 & 3 \\ 2 & 7 \end{array} \right] \\ \left[\begin{array}{cc} 2 & 0 \\ 0 & 2 \end{array} \right] & \left[\begin{array}{cc} 8 & 6 \\ 4 & 12 \end{array} \right] \end{array} \right]$$

$$E_{4,4} = \begin{bmatrix} 2 & 2 & 5 & 3 \\ 3 & 5 & 2 & 7 \\ 2 & 0 & 8 & 6 \\ 0 & 2 & 4 & 12 \end{bmatrix}$$

On peut considérer E soit comme une matrice 2x2 dont les éléments sont des matrices 2x2, soit 4x4 avec des éléments plus classiques : les nombres. L'utilisation de telle ou telle représentation d'un même résultat final dépendra du traitement qu'on a à faire subir à la structure.

2.1.2 Généralisation

On ne peut définir l'addition que si :

- Les matrices (collections ordonnées d'objets) ont la même structure (même nombre de lignes et même nombre de colonnes).
- Il existe une opération d'addition entre les éléments de ces structures (dans le cas des nombres c'est évident mais dans une définition plus complexe de collection d'objets autre cela l'est moins).

$$\left. \begin{array}{l} A_{l_1, c_1} = \left[\left\{ a_{i,j} \right\} \right]_{l_1, c_1} \\ B_{l_2, c_2} = \left[\left\{ b_{i,j} \right\} \right]_{l_2, c_2} \end{array} \right\} A_{l_1, c_1} + B_{l_2, c_2} = C_{l_3, c_3} = \left[\left\{ c_{i,j} \right\} \right]_{l_3, c_3} \text{ avec } \begin{cases} l_1 = l_2 = l_3 \\ c_1 = c_2 = c_3 \\ \forall i \in [1, l_1] \text{ et } \forall j \in [1, c_1], c_{i,j} = a_{i,j} + b_{i,j} \end{cases}$$

- La somme est commutative si la somme des éléments l'est. Dans le cas des nombres et sous matrices de nombres (les éléments sont des matrices –si si c'est possible, C.F. exemple– et dans ce cas on a un processus récursif !!) cela est vrai, et $A_{l,c} + B_{l,c} = B_{l,c} + A_{l,c}$. Nota : normalement, dans la vie de tous les jours où apparaissent ces structures matricielles, (ce qui n'est pas forcément tous les jours) cela est vrai. Dans notre société où tout fini par se représenter par une quantité elle-même représentée par un nombre, on s'appuie sur l'arithmétique des nombres pour qui l'opération de somme est commutative.

2.2 Multiplication

2.2.1 Exemple

Faisons simple

$$A = \begin{bmatrix} 1 & 3 \end{bmatrix}; B = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

$$A \times B = \begin{bmatrix} 1 & 3 \end{bmatrix} \times \begin{bmatrix} 2 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 \cdot 2 + 3 \cdot 5 \end{bmatrix} = \begin{bmatrix} 17 \end{bmatrix}$$

$$B \times A = \begin{bmatrix} 2 \\ 5 \end{bmatrix} \times \begin{bmatrix} 1 & 3 \end{bmatrix} = \begin{bmatrix} 2 \cdot 1 & 2 \cdot 3 \\ 5 \cdot 1 & 5 \cdot 3 \end{bmatrix} = \begin{bmatrix} 2 & 6 \\ 5 & 15 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}; B = \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix}$$

$$A \times B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \cdot 4 + 2 \cdot 2 & 1 \cdot 3 + 2 \cdot 1 \\ 3 \cdot 4 + 4 \cdot 2 & 3 \cdot 3 + 4 \cdot 1 \end{bmatrix} = \begin{bmatrix} 8 & 5 \\ 20 & 13 \end{bmatrix}$$

$$B \times A = \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 4 \cdot 1 + 3 \cdot 3 & 4 \cdot 2 + 3 \cdot 4 \\ 2 \cdot 1 + 1 \cdot 3 & 2 \cdot 2 + 1 \cdot 4 \end{bmatrix} = \begin{bmatrix} 13 & 20 \\ 5 & 8 \end{bmatrix} \neq A \times B$$

On peut d'ordre et déjà remarquer que le produit n'est pas commutatif (un contre exemple suffit à invalider un postulat !).

Généralisons un peu en utilisant les indices :

Sous exemple 1 :

$$A_{4,1} = \begin{bmatrix} a_{1,1} \\ a_{2,1} \\ a_{3,1} \\ a_{4,1} \end{bmatrix}; B_{1,4} = [b_{1,1} \quad b_{1,2} \quad b_{1,3} \quad b_{1,4}]$$

$$A_{4,1} \times B_{1,4} = \begin{bmatrix} a_{1,1} \\ a_{2,1} \\ a_{3,1} \\ a_{4,1} \end{bmatrix} \times [b_{1,1} \quad b_{1,2} \quad b_{1,3} \quad b_{1,4}] = \begin{bmatrix} a_{1,1} \cdot b_{1,1} & a_{1,1} \cdot b_{1,2} & a_{1,1} \cdot b_{1,3} & a_{1,1} \cdot b_{1,4} \\ a_{2,1} \cdot b_{1,1} & a_{2,1} \cdot b_{1,2} & a_{2,1} \cdot b_{1,3} & a_{2,1} \cdot b_{1,4} \\ a_{3,1} \cdot b_{1,1} & a_{3,1} \cdot b_{1,2} & a_{3,1} \cdot b_{1,3} & a_{3,1} \cdot b_{1,4} \\ a_{4,1} \cdot b_{1,1} & a_{4,1} \cdot b_{1,2} & a_{4,1} \cdot b_{1,3} & a_{4,1} \cdot b_{1,4} \end{bmatrix}$$

$$A_{4,1} \times B_{1,4} = D_{4,4} \text{ avec } d_{i,j} = a_{i,1} \cdot b_{1,j} = \sum_{n=1}^{n=1} a_{i,n} \cdot b_{n,j}$$

Ou encore dans l'autre sens :

$$A_{1,4} = [a_{1,1} \quad a_{1,2} \quad a_{1,3} \quad a_{1,4}]; B_{4,1} = \begin{bmatrix} b_{1,1} \\ b_{2,1} \\ b_{3,1} \\ b_{4,1} \end{bmatrix}$$

$$A_{1,4} \times B_{4,1} = [a_{1,1} \quad a_{1,2} \quad a_{1,3} \quad a_{1,4}] \times \begin{bmatrix} b_{1,1} \\ b_{2,1} \\ b_{3,1} \\ b_{4,1} \end{bmatrix} = [a_{1,1} \cdot b_{1,1} + a_{1,2} \cdot b_{2,1} + a_{1,3} \cdot b_{3,1} + a_{1,4} \cdot b_{4,1}]$$

$$A_{1,4} \times B_{4,1} = C_{1,1} \text{ avec } c_{1,1} = \sum_{n=1}^{n=4} a_{1,n} \cdot b_{n,1}$$

On reconnaîtra ici le produit scalaire de deux vecteurs :

$$A_{4,1} = \begin{bmatrix} a_{1,1} \\ a_{2,1} \\ a_{3,1} \\ a_{4,1} \end{bmatrix}; B_{4,1} = \begin{bmatrix} b_{1,1} \\ b_{2,1} \\ b_{3,1} \\ b_{4,1} \end{bmatrix}$$

$$A_{4,1} \bullet B_{4,1} = A_{4,1}^T \times B_{4,1} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \end{bmatrix} \times \begin{bmatrix} b_{1,1} \\ b_{2,1} \\ b_{3,1} \\ b_{4,1} \end{bmatrix} = \begin{bmatrix} a_{1,1} \cdot b_{1,1} + a_{1,2} \cdot b_{2,1} + a_{1,3} \cdot b_{3,1} + a_{1,4} \cdot b_{4,1} \end{bmatrix}$$

$A_{4,1} \bullet B_{4,1} = C_{1,1}$ avec $c_{1,1} = \sum_{n=1}^{n=4} a_{1,n} \cdot b_{n,1}$ $A_{4,1}^T$ est la transposée de $A_{4,1}$ c'est-à-dire qu'il y a pour passer de l'une à l'autre, inversion des lignes et colonnes $\{a_{i,j}^T = a_{j,i}\}$.

Sous exemple 2 :

$$A_{2,4} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \end{bmatrix}; B_{4,1} = \begin{bmatrix} b_{1,1} \\ b_{2,1} \\ b_{3,1} \\ b_{4,1} \end{bmatrix} \text{ et donc } A_{2,4} \times B_{4,1} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \end{bmatrix} \times \begin{bmatrix} b_{1,1} \\ b_{2,1} \\ b_{3,1} \\ b_{4,1} \end{bmatrix}$$

$$= \begin{bmatrix} a_{1,1} \cdot b_{1,1} + a_{1,2} \cdot b_{2,1} + a_{1,3} \cdot b_{3,1} + a_{1,4} \cdot b_{4,1} \\ a_{2,1} \cdot b_{1,1} + a_{2,2} \cdot b_{2,1} + a_{2,3} \cdot b_{3,1} + a_{2,4} \cdot b_{4,1} \end{bmatrix} = C_{2,1}; A_{2,4} \times B_{4,1} = C_{2,1} \text{ avec } c_{i,j} = \sum_{n=1}^{n=4} a_{i,n} \cdot b_{n,j}$$

dans l'autre sens :

$$A_{4,2} = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \\ a_{4,1} & a_{4,2} \end{bmatrix}; B_{2,1} = \begin{bmatrix} b_{1,1} \\ b_{2,1} \end{bmatrix}$$

$$A_{4,2} \times B_{2,1} = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \\ a_{4,1} & a_{4,2} \end{bmatrix} \times \begin{bmatrix} b_{1,1} \\ b_{2,1} \end{bmatrix} = \begin{bmatrix} a_{1,1} \cdot b_{1,1} + a_{1,2} \cdot b_{2,1} \\ a_{2,1} \cdot b_{1,1} + a_{2,2} \cdot b_{2,1} \\ a_{3,1} \cdot b_{1,1} + a_{3,2} \cdot b_{2,1} \\ a_{4,1} \cdot b_{1,1} + a_{4,2} \cdot b_{2,1} \end{bmatrix} = D_{4,1}$$

$$A_{4,2} \times B_{2,1} = D_{4,1} \text{ avec } d_{i,j} = \sum_{n=1}^{n=2} a_{i,n} \cdot b_{n,j}$$

sous exemple 3 :

$$A_{2,4} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \end{bmatrix}; B_{4,2} = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \\ b_{4,1} & b_{4,2} \end{bmatrix}$$

$$A_{2,4} \times B_{4,2} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \end{bmatrix} \times \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \\ b_{4,1} & b_{4,2} \end{bmatrix}$$

$$= \begin{bmatrix} a_{1,1} \cdot b_{1,1} + a_{1,2} \cdot b_{2,1} + a_{1,3} \cdot b_{3,1} + a_{1,4} \cdot b_{4,1} & a_{1,1} \cdot b_{1,2} + a_{1,2} \cdot b_{2,2} + a_{1,3} \cdot b_{3,2} + a_{1,4} \cdot b_{4,2} \\ a_{2,1} \cdot b_{1,1} + a_{2,2} \cdot b_{2,1} + a_{2,3} \cdot b_{3,1} + a_{2,4} \cdot b_{4,1} & a_{2,1} \cdot b_{1,2} + a_{2,2} \cdot b_{2,2} + a_{2,3} \cdot b_{3,2} + a_{2,4} \cdot b_{4,2} \end{bmatrix} = C_{2,2}$$

$$A_{2,4} \times B_{4,2} = C_{2,2} \text{ avec } c_{i,j} = \sum_{n=1}^{n=4} a_{i,n} \cdot b_{n,j}$$

Dans l'autre sens :

$$A_{4,2} = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \\ a_{4,1} & a_{4,2} \end{bmatrix}; B_{2,4} = \begin{bmatrix} b_{1,1} & b_{1,2} & b_{1,3} & b_{1,4} \\ b_{2,1} & b_{2,2} & b_{2,3} & b_{2,4} \end{bmatrix}$$

$$D_{4,4} = A_{4,2} \times B_{2,4} = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \\ a_{4,1} & a_{4,2} \end{bmatrix} \times \begin{bmatrix} b_{1,1} & b_{1,2} & b_{1,3} & b_{1,4} \\ b_{2,1} & b_{2,2} & b_{2,3} & b_{2,4} \end{bmatrix}$$

$$= \begin{bmatrix} a_{1,1} \cdot b_{1,1} + a_{1,2} \cdot b_{2,1} & a_{1,1} \cdot b_{1,2} + a_{1,2} \cdot b_{2,2} & a_{1,1} \cdot b_{1,3} + a_{1,2} \cdot b_{2,3} & a_{1,1} \cdot b_{1,4} + a_{1,2} \cdot b_{2,4} \\ a_{2,1} \cdot b_{1,1} + a_{2,2} \cdot b_{2,1} & a_{2,1} \cdot b_{1,2} + a_{2,2} \cdot b_{2,2} & a_{2,1} \cdot b_{1,3} + a_{2,2} \cdot b_{2,3} & a_{2,1} \cdot b_{1,4} + a_{2,2} \cdot b_{2,4} \\ a_{3,1} \cdot b_{1,1} + a_{3,2} \cdot b_{2,1} & a_{3,1} \cdot b_{1,2} + a_{3,2} \cdot b_{2,2} & a_{3,1} \cdot b_{1,3} + a_{3,2} \cdot b_{2,3} & a_{3,1} \cdot b_{1,4} + a_{3,2} \cdot b_{2,4} \\ a_{4,1} \cdot b_{1,1} + a_{4,2} \cdot b_{2,1} & a_{4,1} \cdot b_{1,2} + a_{4,2} \cdot b_{2,2} & a_{4,1} \cdot b_{1,3} + a_{4,2} \cdot b_{2,3} & a_{4,1} \cdot b_{1,4} + a_{4,2} \cdot b_{2,4} \end{bmatrix}$$

$$A_{4,2} \times B_{2,4} = D_{4,4} \text{ avec } d_{i,j} = \sum_{n=1}^{n=2} a_{i,n} \cdot b_{n,j}$$

2.2.2 Généralisation :

Cas de matrices quelconques (pas tout à fait cependant)

$$A_{i,j} \times B_{k,l} = C_{i,l} \text{ avec } \begin{cases} i = l \\ j = k \\ \forall p \in [1, i]; \forall q \in [1, j]; c_{p,q} = \sum_{n=1}^{n=i} a_{p,n} \cdot b_{n,q} \end{cases}$$

$$\text{Cas des matrices carrées : } A_{p,p} \times B_{p,p} = C_{p,p} \text{ avec } c_{i,j} = \sum_{n=1}^{n=p} a_{i,n} \cdot b_{n,j}$$

$$\text{Produit d'une matrice carrée par un vecteur : } A_{n,n} \times B_n = C_n \text{ avec } c_n = \sum_{i=1}^{i=n} a_{n,i} \cdot b_i$$

On ne peut définir la multiplication de deux matrices que si :

- Les matrices (collections ordonnées d'objets) ont une structure compatible i.e. le nombre de colonnes de la première matrice correspond au nombre de lignes de la deuxième matrice.
- Il existe une opération de multiplication entre les éléments de ces structures (dans le cas des nombres c'est évident mais dans une définition plus complexe de collection d'objets autre cela l'est moins)

$$\left. \begin{array}{l} A_{p,q} = \left[\left\{ a_{i,j} \right\} \right] \\ B_{q,r} = \left[\left\{ b_{i,j} \right\} \right] \end{array} \right\} A_{p,q} \times B_{q,r} = C_{p,r} = \left[\left\{ c_{i,j} \right\} \right]_{p,r} \text{ avec } \forall i \in [1, p] \text{ et } \forall j \in [1, r], c_{i,j} = \sum_{n=1}^{n=p} a_{i,n} \cdot b_{n,j}$$

La multiplication n'est pas commutative !

Comme pour l'addition, on peut définir la multiplication de matrices à condition que les sous éléments des matrices puissent se multiplier entre eux. Dans le cas de matrices constituées d'éléments matrices carrées identiques, cela ne pose pas de problème, mais dans le cas de sous éléments composites rectangulaires, la question reste à étudier au cas par cas.

Exemple :

$$\begin{bmatrix} A_{3,3} & B_{3,1} \\ C_{1,3} & D_{1,1} \end{bmatrix} \times \begin{bmatrix} \widetilde{A}_{3,3} & \widetilde{B}_{3,1} \\ \widetilde{C}_{1,3} & \widetilde{D}_{1,1} \end{bmatrix} = \begin{bmatrix} A_{3,3} \times \widetilde{A}_{3,3} + B_{3,1} \times \widetilde{C}_{1,3} & A_{3,3} \times \widetilde{B}_{3,1} + B_{3,1} \times \widetilde{D}_{1,1} \\ C_{1,3} \times \widetilde{A}_{3,3} + D_{1,1} \times \widetilde{C}_{1,3} & C_{1,3} \times \widetilde{B}_{3,1} + D_{1,1} \times \widetilde{D}_{1,1} \end{bmatrix}$$

Dans ce cas on pourra vérifier que les composantes de la matrice résultant de la multiplication sont valides et que leur type correspond au type des matrices initiales.

$$\begin{cases} A_{3,3} \times \widetilde{A}_{3,3} + B_{3,1} \times \widetilde{C}_{1,3} \text{ est du type carré } 3 \times 3 \\ A_{3,3} \times \widetilde{B}_{3,1} + B_{3,1} \times \widetilde{D}_{1,1} \text{ est du type vecteur } 3 \times 1 \\ C_{1,3} \times \widetilde{A}_{3,3} + D_{1,1} \times \widetilde{C}_{1,3} \text{ est du type ligne } 1 \times 3 \\ C_{1,3} \times \widetilde{B}_{3,1} + D_{1,1} \times \widetilde{D}_{1,1} \text{ est du type carré } 1 \times 1 \end{cases}$$

Ce type de calcul pourra être effectué lorsque la décomposition des matrices initiales en sous matrices produit des sous matrices particulières (nulles, unités ou identités) qui permettent de connaître sans calcul le résultat des sous produits résultants et donc de limiter les calculs (le temps c'est de l'argent même en informatique !!).

2.3 Multiplication par un nombre (élément).

2.3.1 Exemple

Faisons simple

$$A = \begin{bmatrix} 1 & 3 \end{bmatrix}; B = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

$$1,5 \cdot A = 1,5 \cdot \begin{bmatrix} 1 & 3 \end{bmatrix} = \begin{bmatrix} 1,5 \cdot 1 & 1,5 \cdot 3 \end{bmatrix} = \begin{bmatrix} 1,5 & 4,5 \end{bmatrix}$$

$$1,5 \cdot B = 1,5 \cdot \begin{bmatrix} 2 \\ 5 \end{bmatrix} = \begin{bmatrix} 1,5 \cdot 2 \\ 1,5 \cdot 5 \end{bmatrix} = \begin{bmatrix} 3 \\ 7,5 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}; B = \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix}$$

$$1,5 \cdot A = \begin{bmatrix} 1,5 \cdot 1 & 1,5 \cdot 2 \\ 1,5 \cdot 3 & 1,5 \cdot 4 \end{bmatrix} = \begin{bmatrix} 1,5 & 3 \\ 4,5 & 6 \end{bmatrix}$$

2.3.2 Généralisation :

La multiplication dite externe d'une matrice par un élément se traduit par la multiplication de tous les termes de la matrice par ce même élément

$$A_{p,q} = \left[\{ a_{i,j} \} \right]_{p,q}; \lambda \cdot A_{p,q} = C_{p,q} = \left[\{ c_{i,j} \} \right]_{p,q} \text{ avec } \forall i \in [1, p] \text{ et } \forall j \in [1, q], c_{i,j} = \lambda \cdot a_{i,j}.$$

Nota :

Si $\lambda = \lambda_{k,l}$ est une matrice, elle doit pouvoir se multiplier avec tous les éléments de la matrice A . On pourra donc utiliser cette multiplication particulière dans le cas où les sous éléments de la matrice A sont des matrices carrées, et que le multiplicateur $\lambda = \lambda_{k,k}$ est de la même taille que les sous éléments.

La multiplication externe est en fait une fausse multiplication de matrices standards :

$$A_{p,q} = \left[\{a_{i,j}\} \right]_{p,q}; \lambda \cdot A_{p,q} = \begin{bmatrix} \lambda & 0 & \dots & 0 \\ \lambda & 0 & \dots & 0 \\ \dots & \dots & \dots & 0 \\ \lambda & 0 & \dots & 0 \end{bmatrix}_{q,p} \times A_{p,q} = C_{p,q} = \left[\{c_{i,j}\} \right]_{p,q} \text{ avec } \forall i \in [1, p] \text{ et } \forall j \in [1, q], c_{i,j} = \lambda \cdot a_{i,j}$$

On peut donc envisager comme les autres de traiter de manière automatique ce type de multiplications comme les autres (mais en terme d'efficacité de calcul on est loin d'être optimum).

2.4 Déterminant d'une matrice

Le déterminant d'une matrice n'est pas à proprement parler une opération. C'est un nombre caractéristique calculé à partir des coefficients de la matrice. On retrouve ce nombre dans beaucoup d'opérations sur les matrices et dans l'utilisation de la représentation matricielle dans la résolution des systèmes d'équation linéaires.

2.4.1 Exemple

Dans l'exemple suivant, outre la méthode de calcul, nous nous présenterons les différentes notations qui nous seront nécessaires par la suite.

$$A = [1]; \det(A) = |A| = 1$$

$$B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}; \det(A) = \begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} = 1 \times 4 - 3 \times 2 = -2$$

$$C = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}; \det(C) = \begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix}$$

$$\det(C) = (-1)^{(1+1)} \times 1 \times \det \left(\begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix} \right) + (-1)^{(1+2)} \times 2 \times \det \left(\begin{bmatrix} 4 & 6 \\ 7 & 9 \end{bmatrix} \right) + (-1)^{(1+3)} \times 3 \times \det \left(\begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix} \right) = 0$$

$$\det(C) = (-1)^{(1+1)} \times 1 \times \det(Co(1,1)) + (-1)^{(1+2)} \times 2 \times \det(Co(1,2)) + (-1)^{(1+3)} \times 3 \times \det(Co(1,3)) = 0$$

$$\det(C) = (-1)^{(1+1)} \times 1 \times |Co(1,1)| + (-1)^{(1+2)} \times 2 \times |Co(1,2)| + (-1)^{(1+3)} \times 3 \times |Co(1,3)| = 0$$

$$\text{avec } Co(1,1)_{2,2} = \begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}; Co(1,2)_{2,2} = \begin{bmatrix} 4 & 6 \\ 7 & 9 \end{bmatrix}; Co(1,3)_{2,2} = \begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix} \text{ les co-matrices de la matrice initiale.}$$

$$|C| = 1 \times (-3) - 2 \times (-6) + 3 \times (-3) = 0$$

Vous Vous aurez compris, la notation $[A]$ représente la matrice et $|A|$ son déterminant.

2.4.2 Généralisation

La détermination du déterminant d'une matrice de manière classique est un processus « récursif ». On développe le déterminant suivant une ligne (ou une colonne) et on calcule les déterminants des *co-matrices* associées. Les *co-matrices* sont des matrices d'ordre inférieur de 1 à la matrice initiale et qui contiennent les mêmes termes que la matrice initiale, à l'exception des termes de la ligne et de la colonne correspondant au terme calculé :

$$\text{avec } A_{n,n} = \left[\{a_{i,j}\} \right]_{n,n}$$

en développant sur une ligne l , $\det(A_{n,n}) = |A_{n,n}| = \sum_{c=1}^{c=n} \left((-1)^{(l+c)} a_{l,c} \times |C(l,c)_{n-1,n-1}| \right); l \in [1, n]$

en développant suivant une colonne c , $\det(A_{n,n}) = |A_{n,n}| = \sum_{l=1}^{l=n} \left((-1)^{(l+c)} a_{l,c} \times |C(l,c)_{n-1,n-1}| \right); c \in [1, n]$

la comatrice C de rang $(n-1) \times (n-1)$ est définie comme tel :

$$C(l,c)_{m,m} = [c_{p,q}]_{m,m} \text{ à partir de } A_{n,n} = [a_{i,j}]$$

$$\text{telle que } \begin{cases} \left\{ \begin{array}{l} p = i \text{ si } p < l \\ p = i + 1 \text{ si } p > l \end{array} \right. \\ \left\{ \begin{array}{l} q = j \text{ si } q < c \\ q = j + 1 \text{ si } q > c \end{array} \right. \end{cases} \text{ et } |C(l,c)_{n-1,n-1}| \text{ le déterminant de cette comatrice.}$$

2.4.3 Propriétés du déterminant

Nous pouvons montrer que :

$$\det(B \times A) = \det(A \times B) = \det(A) \cdot \det(B)$$

$$\det(I) = 1$$

Si B est obtenu en inversant 2 lignes ou 2 colonnes de A alors, $\det(B) = -\det(A)$. Lorsque dans des manipulations futures nous effectuerons n inversions, nous aurons $\det(B) = (-1)^n \det(A)$.

La multiplication d'une ligne de la matrice par un coef k provoque la multiplication du déterminant par ce même coef. k. Il en résulte que $\det(\lambda \cdot B_{n,n}) = \lambda^n \det(B_{n,n})$.

Lorsque qu'on ajoute à une ligne d'une matrice une combinaison linéaire d'autre ligne de la même matrice, le déterminant ne change pas.

La transposée d'une matrice de rang n pouvant être obtenue par *troncatureBasse*(n/2) (ex pour n=3, 1 inversion, pour n=4, 2 inversions ...) inversions de lignes suivie par le même nombre d'inversions de colonnes,

$$\det(A^T) = \det(A)$$

Une propriété de la matrice des déterminants des cofacteurs :

$$A^{-1} = \frac{1}{|A|} C^T \text{ avec } C \text{ la matrice des déterminants des cofacteurs de } A$$

$$C = [c_{i,j}] \text{ avec } c_{i,j} = |Co_A(i, j)|$$

$$\det(C^T) = \det(C) = 1$$

$$\det(A^{-1}) = \frac{1}{\det(A)}$$

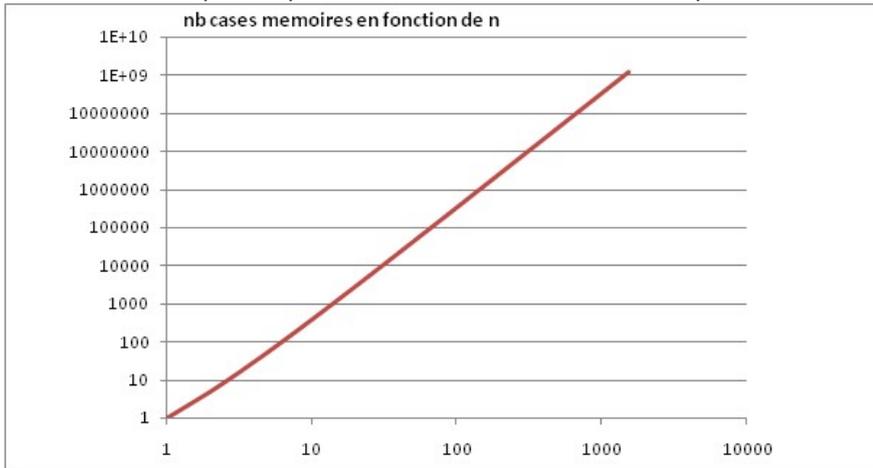
Si M est une matrice diagonale ou trigonale, avec $M = [m_{i,j}]_{n,n}$; $\det(M) = \prod_{i=1}^n m_{i,i}$

2.4.4 Efficacité du calcul du déterminant

Le calcul du déterminant par cette méthode (historique) est gourmand en temps et en mémoire. Le nombre de

valeurs réelles en mémoire nécessaires pour le calcul d'un déterminant d'une matrice $n \times n$ est de $t = \sum_{i=1}^n i^2$. Ceci

croit moins vite que l'exponentielle mais tout de même rapidement.



Cet état de fait nous amène à d'autres méthodes basées sur des « ruses » pour déterminer les déterminants de grosses matrices utilisées dans les calculs de simulation. Par exemple en Météo, en maillant le terrain tous les km en x et y et tous les 100m en z, pour une simulation standard de température-pression sur la France (1500 km par 1500 km par 8000m) il nous faut une matrice de $(1500 \times 1500 \times 80)$ coefficients soit 180 000 000. Oui vous avez bien lu ! 180 millions ! à raison de 8 octets pour coder un coefficient, cela conduit à une mémoire vive de 1,357Go ! Sans parler du temps calcul bien évidemment.

L'algorithme de calcul classique hautement récursif est donc :

Algorithme	Exemple d'implémentation en C++	Analyse de récursivité
<p>Déterminant de $A = \left[\left\{ a_{i,j} \right\} \right]$</p> <p>si taille=1x1 Alors renvoyer $a_{1,1}$</p> <p>sinon deter=0</p> <p>Prendre la première ligne de la matrice Du premier élément au dernier élément faire Créer la co-matrice C correspondante Calculer son déterminant $d = C$ deter = deter + $a_{1,colonne} \times (-1)^{colonne+1} \times d$</p> <p>renvoyer deter</p>	<pre> real determinant(real* M, int size) { #define M(x,y) M[(x)*size+(y)] #define C(x,y) C[(x)*size+(y)] if (size == 1) return M[0]; real deter = 0; for (int col = 0; col < size; col++) { real C[(size - 1) * (size - 1)]; int nc = 0; for (int c = 0; c < size; c++) { if (c != col) { for (int l = 1; l < size; l++) C[(l-1),nc] = M(l,c); nc++; } } if (col % 2 == 0) deter += M[col] * determinant(C, size - 1); else deter -= M[0,col] * determinant(C, size - 1); } return deter; #undef M #undef C } </pre>	<p>avec $C_B(1,c)_{r,r}$ la matrice des cofacteurs de B; c : colonne, r : taille de la matrice, p : profondeur de récursivité</p>

Le calcul du déterminant d'une matrice 4x4 nécessite le calcul de 4 déterminants 3x3 avec pour chacun 3

déterminants 2x2 qui se calculent eux-mêmes à l'aide de 2 déterminants 1x1.

La profondeur de récursivité atteinte est donc de 3. Les déterminants 3x3, 2x2 et 1x1 étant « triviaux », il est possible de diminuer les appels récursifs au niveau 3x3.

III Matrices particulières

3.1 Matrice unité

I tel que $A \times I = I \times A = A$

$$I_{n,n} = \left[\begin{array}{cccccc} \overbrace{1 & 0 & \dots & 0 & 0}^{n \text{ colonnes}} \\ 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{array} \right] \left. \vphantom{\begin{array}{c} \\ \\ \\ \\ \end{array}} \right\} n \text{ lignes avec } i_{p,q} = \text{SI}(p == q) \{1\} \text{ SINON } \{0\}$$

3.2 Transposée d'une matrice

Il s'agit « d'inverser » les lignes et les colonnes

Exemple $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$

$$A_{n,n} = [a_{i,j}]; B_{n,n} = [b_{i,j}] = A_{n,n}^T \Rightarrow b_{i,j} = a_{j,i} \quad \forall i \in [1..n], \forall j \in [1..n]$$

On notera les propriétés suivantes

$$(A^T)^T = A ; (A \times B)^T = B^T \times A^T$$

3.3 L'inverse d'une matrice

Définition

$$A^{-1} \text{ tel que } A \times A^{-1} = A^{-1} \times A = I$$

Calcul

Le calcul de l'inverse d'une matrice peut se faire à l'aide des déterminants par la formule

$$A^{-1} = \frac{1}{|A|} C^T \text{ avec } C \text{ la matrice des déterminants des cofacteurs de } A$$

$C = [c_{i,j}]$ avec $c_{i,j} = (-1)^{i+j} |Co_A(i, j)|$. Les cofacteurs de $A_{n,n}$ sont des matrices $Co_{n-1,n-1}(i, j)$ dont les coefficients sont tous les coefficients de A à l'exception de ceux de la ligne i et de la colonne j . $co_{k,l} = a_{p,q}$ avec $\begin{cases} \text{si } k < i \text{ alors } k = p \\ \text{si } k > i \text{ alors } k = p + 1 \end{cases}$ et $\begin{cases} \text{si } l < j \text{ alors } l = q \\ \text{si } l > j \text{ alors } l = q + 1 \end{cases}$.

Le calcul par cette méthode est possible mais conduit à une débauche de calcul et de mémoire. En effet pour $A_{n,n}$ nous devons calculer : un déterminant d'ordre n et n^2 déterminants d'ordre $n-1$.

Si on utilise la procédure classique de calcul des déterminants et que la récursivité s'arrête au niveau 2x2, le

nombre de déterminants 2x2 à calculer sera de $\frac{n!}{2} + n^2 \frac{(n-1)!}{2} = \frac{(n-1)!}{2} \cdot (n + n^2) = p$. Astronomique non ?!

Pour une matrice 10×10 $p = \frac{9!}{2} \cdot (10 + 10^2) = 19\,958\,400$ déterminant 2×2 soit approximativement 60 000 000 opérations flottantes. Avec un processeur *Intel Core Duo T9500* qui développe 20 GFLOP, l'opération durera $60/2000=0.03$ s mais avec un microcontrôleur du style 328p à 16 Mhz qui développe une puissance proche de 0.1 MFLOP cette opération dure $60/0.1=600$ s= 10 mn. Il est clair alors qu'il vaut mieux passer son chemin ou utiliser un algorithme plus efficace.

De plus comme nous l'avons vu précédemment, la mémoire utilisée pour calculer un déterminant 10×10 est de 385 cases mémoires à raison de 4 octets par case (simple précision). Nous atteignons alors les 1540 octets soit 1.5 Ko (la mémoire vive de ces microcontrôleurs est de 2Ko !)

Inverse d'un produit de matrices : On peut montrer que $(A \times B)^{-1} = B^{-1} \times A^{-1}$

Exemple pour s'en persuader : Dans le bâtiment, pour finir un mur, on l'enduit (opération A) puis on le peint

(opération B) soit la transformation $\begin{pmatrix} mur \\ fini \end{pmatrix} = B \times \left(A \times \begin{pmatrix} mur \\ brut \end{pmatrix} \right) = B \times A \times \begin{pmatrix} mur \\ brut \end{pmatrix} = (B \times A) \times \begin{pmatrix} mur \\ brut \end{pmatrix}$. Pour

retrouver un mur brut à partir du mur fini, on enlève la peinture B^{-1} puis on enlève l'enduit A^{-1}

$$\begin{pmatrix} mur \\ brut \end{pmatrix} = A^{-1} \times \left(B^{-1} \times \begin{pmatrix} mur \\ fini \end{pmatrix} \right) = A^{-1} \times B^{-1} \begin{pmatrix} mur \\ fini \end{pmatrix} = (B \times A)^{-1} \begin{pmatrix} mur \\ fini \end{pmatrix}$$

Inverse de la somme de matrices :

Ah là il n'y a pas de solutions simples ... on peut montrer que si $A + B$ est inversible alors (développement de Neumann) :

$$(A + B)^{-1} = A^{-1} \times A \times (A + B)^{-1} = A^{-1} \times (A^{-1} \times A + A^{-1} \times B)^{-1} = A^{-1} \times (I + A^{-1} \times B)^{-1}.$$

$$(A + B)^{-1} = A^{-1} \times \sum_{n=0}^{\infty} \left((-1)^n (A^{-1} \times B)^n \right)$$

Le résultat précédent est juste là pour les puristes et pour montrer la non évidence de cette opération.

3.4 Matrice trigonale supérieure (U) up

$$A_{n,n} = \left[\begin{array}{cccccc} & & & & \overbrace{\hspace{2cm}} & \\ & & & & n \text{ colonnes} & \\ a_{1,1} & a_{1,2} & \dots & a_{1,n-1} & a_{1,n} & \\ 0 & a_{2,1} & \dots & a_{2,n-1} & a_{2,n} & \\ \dots & \dots & \dots & \dots & \dots & \\ 0 & 0 & \dots & a_{n-1,n-1} & a_{n-1,n} & \\ 0 & 0 & \dots & 0 & a_{n,n} & \end{array} \right] \left. \vphantom{\begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array}} \right\} n \text{ lignes avec } a_{p,q} = \text{SI}(p > q) \{0\}$$

3.5 Matrice trigonale inférieure (L) low

$$A_{n,n} = \left[\begin{array}{cccccc} & & & & \overbrace{\hspace{2cm}} & \\ & & & & n \text{ colonnes} & \\ a_{1,1} & 0 & \dots & 0 & 0 & \\ a_{2,1} & a_{2,1} & \dots & 0 & 0 & \\ \dots & \dots & \dots & \dots & \dots & \\ a_{n-1,1} & a_{n-1,2} & \dots & a_{n-1,n-1} & 0 & \\ a_{n,1} & a_{n,2} & \dots & a_{n,n-1} & a_{n,n} & \end{array} \right] \left. \vphantom{\begin{array}{c} \\ \\ \\ \\ \\ \end{array}} \right\} n \text{ lignes avec } a_{p,q} = \text{SI}(p < q) \{0\}$$

3.6 Matrice diagonale

$$A_{n,n} = \left[\begin{array}{ccccc} a_{1,1} & 0 & \dots & 0 & 0 \\ 0 & a_{2,1} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_{n-1,n-1} & 0 \\ 0 & 0 & \dots & 0 & a_{n,n} \end{array} \right] \left. \vphantom{\begin{array}{ccccc} a_{1,1} & 0 & \dots & 0 & 0 \\ 0 & a_{2,1} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_{n-1,n-1} & 0 \\ 0 & 0 & \dots & 0 & a_{n,n} \end{array}} \right\} n \text{ lignes avec } a_{p,q} = \text{SI}(p \neq q) \{0\}$$

Puissance d'une matrice diagonale :

$$A_{n,n} = \left[\begin{array}{ccccc} a_{1,1} & 0 & \dots & 0 & 0 \\ 0 & a_{2,2} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_{n-1,n-1} & 0 \\ 0 & 0 & \dots & 0 & a_{n,n} \end{array} \right] \left. \vphantom{\begin{array}{ccccc} a_{1,1} & 0 & \dots & 0 & 0 \\ 0 & a_{2,2} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_{n-1,n-1} & 0 \\ 0 & 0 & \dots & 0 & a_{n,n} \end{array}} \right\} n \text{ lignes et donc } (A_{n,n})^p = \left[\begin{array}{ccccc} (a_{1,1})^p & 0 & \dots & 0 & 0 \\ 0 & (a_{2,2})^p & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & (a_{n-1,n-1})^p & 0 \\ 0 & 0 & \dots & 0 & (a_{n,n})^p \end{array} \right]$$

Inverse d'une matrice diagonale :

$$A_{n,n} = \left[\begin{array}{ccccc} a_{1,1} & 0 & \dots & 0 & 0 \\ 0 & a_{2,2} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_{n-1,n-1} & 0 \\ 0 & 0 & \dots & 0 & a_{n,n} \end{array} \right] \left. \vphantom{\begin{array}{ccccc} a_{1,1} & 0 & \dots & 0 & 0 \\ 0 & a_{2,2} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_{n-1,n-1} & 0 \\ 0 & 0 & \dots & 0 & a_{n,n} \end{array}} \right\} n \text{ lignes}$$

$$A \times A^{-1} = I \text{ fourni } \begin{cases} n \text{ équations du style } \widetilde{a}_{i,i} \cdot a_{i,i} = 1 \text{ et donc } \widetilde{a}_{i,i} = \frac{1}{a_{i,i}} \\ n^2 - n \text{ équations du style } a_{p,p} \times b_{p,j} = 0 \text{ et donc } b_{p,j} = 0 \end{cases}$$

$$\text{et donc } A_{n,n}^{-1} = \left[\begin{array}{ccccc} \widetilde{a}_{1,1} & \widetilde{b}_{1,2} & \dots & \widetilde{b}_{1,n-1} & \widetilde{b}_{1,n} \\ \widetilde{b}_{2,1} & \widetilde{a}_{2,2} & \dots & \widetilde{b}_{2,n-1} & \widetilde{b}_{2,n} \\ \dots & \dots & \dots & \dots & \dots \\ \widetilde{b}_{n-1,1} & \widetilde{b}_{n-1,2} & \dots & \widetilde{a}_{n-1,n-1} & \widetilde{b}_{n-1,n} \\ \widetilde{b}_{n,1} & \widetilde{b}_{n,2} & \dots & \widetilde{b}_{n,n-1} & \widetilde{a}_{n,n} \end{array} \right] = \left[\begin{array}{ccccc} 1/a_{1,1} & 0 & \dots & 0 & 0 \\ 0 & 1/a_{2,2} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1/a_{n-1,n-1} & 0 \\ 0 & 0 & \dots & 0 & 1/a_{n,n} \end{array} \right]$$

On remarque que les solutions existent si et seulement si tous les $a_{i,i}$ sont non nuls.

IV Résolution d'un système linéaire

La résolution de systèmes d'équations linéaires (de quelques équations à plusieurs milliers) trouve une place de choix dans notre monde où tout est modélisé. Les modèles issus du monde des mathématiques et de la physique (qui ne sont guères des modèles linéaires) sont le plus souvent approximés par morceau avec des sous-modèles linéaires qui essaient par leur segmentation de ne pas s'éloigner trop du modèle complexe, essaient de calquer

avec une imprécision réduite le modèle théorique, qui essaie d'interpréter au mieux la réalité physique (on n'est pas loin de l'homme qui a vu l'homme qui a vu l'homme qui L'ours !).

4.1 Exemple introductif

Si nous reprenons l'exemple du 1.1.1 : Le système

$$\begin{cases} 3x - 2y = 2 \\ z - y = 3 \\ x - 2z = 4 \end{cases} \text{ se traduit par l'équation } M \times X = B \text{ soit } \begin{bmatrix} 3 & -2 & 0 \\ 0 & -1 & 1 \\ 1 & 0 & -2 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} \text{ m. Le bon sens et tout ce que}$$

nous avons déjà vu nous ferait écrire $M^{-1} \times M \times X = M^{-1} \times B$ et donc comme $M^{-1} \times M = I$, et $I \times X = X$ nous obtenons $X = M^{-1} \times B$. Comme nous savons calculer M^{-1} , pas besoin d'aller plus loin LA SOLUTION est là !. En calculant les multiples déterminants (1 de rang 3, 9 de rang 2 soit une 50ème de calculs en virgule flottante), nous

$$\text{arrivons à } M^{-1} = \begin{bmatrix} \frac{1}{2} & -1 & -\frac{1}{2} \\ \frac{1}{4} & -\frac{3}{2} & -\frac{3}{4} \\ \frac{1}{4} & -\frac{1}{2} & -\frac{3}{4} \end{bmatrix} \text{ et } |M| = 4$$

$$M^{-1} \times B = \begin{bmatrix} \frac{1}{2} & -1 & -\frac{1}{2} \\ \frac{1}{4} & -\frac{3}{2} & -\frac{3}{4} \\ \frac{1}{4} & -\frac{1}{2} & -\frac{3}{4} \end{bmatrix} \times \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} -4 \\ -7 \\ -4 \end{bmatrix} = X \text{ et donc } \begin{cases} x = -4 \\ y = -7 \\ z = -4 \end{cases} \text{ (en rajoutant 33 opérations en virgule flottante)}$$

soit au total approximativement 85 opérations en virgule flottante)

4.2 Solution mathématique brute :

La solution mathématique brute sans se soucier de la réalisation (l'intendance suivra ! comme disait un fameux général) est, en écrivant le système d'équation sous forme matricielle :

$$A_{n,n} \times X_{n,1} = B_{n,1} \text{ revient à } A_{n,n}^{-1} \times A_{n,n} \times X_{n,1} = A_{n,n}^{-1} \times B_{n,1}$$

$$\text{soit } A_{n,n}^{-1} \times B_{n,1} = X_{n,1}$$

Certes cela fonctionne mais si nous revenons sur nos considérations de taille mémoire et de nombre d'opérations nécessaires pour effectuer le calcul, nous voyons très vite que cette version de puriste mathématique ne peut convenir dans le cas de gros systèmes ni dans le cas de petits processeurs. De même, la résolution à la main de systèmes même de taille modeste nécessite beaucoup de calculs. Ces lacunes ont rapidement mené grand nombre de mathématiciens à trouver des solutions alternatives pour rendre ces opérations réalisables.

4.3 Methode de Kramer



Gabriel Kramer (1704-1752) mathématicien Suisse (Genevois)

Cette méthode basée sur le calcul de déterminants ne brille pas pour son efficacité numérique. Outre son caractère historique, elle permet de prédire une solution lors de démonstrations mathématiques abstraites et, chose non négligeable, de fournir une solution sous forme d'expression arithmétique (complexe certes mais ...) lorsqu'on doit résoudre un système possédant des paramètres algébriques non définis numériquement.

4.3.1 Effectuons le calcul sur un exemple

$$\begin{bmatrix} 3 & -2 & 0 \\ 1 & -1 & -1 \\ 4 & -2 & -2 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 7 \\ 6 \end{bmatrix}$$

Il est possible de démontrer (mais ici nous l'admettrons !) que

$$x = \frac{\begin{vmatrix} 2 & -2 & 0 \\ 7 & -1 & -1 \\ 6 & -2 & -2 \end{vmatrix}}{\begin{vmatrix} 3 & -2 & 0 \\ 1 & -1 & -1 \\ 4 & -2 & -2 \end{vmatrix}} = \frac{-16}{4} = -4; \quad y = \frac{\begin{vmatrix} 3 & 2 & 0 \\ 1 & 7 & -1 \\ 4 & 6 & -2 \end{vmatrix}}{\begin{vmatrix} 3 & -2 & 0 \\ 1 & -1 & -1 \\ 4 & -2 & -2 \end{vmatrix}} = \frac{-28}{4} = -7; \quad z = \frac{\begin{vmatrix} 3 & -2 & 2 \\ 1 & -1 & 7 \\ 4 & -2 & 6 \end{vmatrix}}{\begin{vmatrix} 3 & -2 & 0 \\ 1 & -1 & -1 \\ 4 & -2 & -2 \end{vmatrix}} = \frac{-16}{4} = -4$$

Vous aurez compris la méthode via le jeu des couleurs utilisé : au numérateur (au dessus de la fraction !) on calcule le déterminant des coefficients des équations auquel on a substitué à la colonne initiale (1 pour la solution n°1, 2 pour la solution N°2 ...) la colonne des constantes.

4.3.2 Généralisation

Les solutions de $A_{n,n} \times X_n = B_n$ sont $x_i = \frac{\det(\widetilde{A}_{n,n}^i)}{\det(A_{n,n})}$ où $\widetilde{A}_{n,n}^i$ est la matrice des coefficients de l'équation dans laquelle on a remplacé la colonne d'indice i par la colonne de valeurs issues de B_n .

4.3.3 Algorithmique.

Nous n'allons pas ici faire un grand exposé sur la méthode utilisée dans le domaine du numérique vu qu'elle repose sur la méthode de calcul du déterminant exposée précédemment.

Cette méthode utilise au moins $n \times n$ cases mémoires de plus que celle utilisée dans le calcul d'un déterminant classique et consomme $(n+1)$ fois plus de temps calcul d'un déterminant (astronomique !!). Nous comprendrons aisément que seuls les mathématiciens déconnectés de la réalité pratique l'utilisent en cours de mathématiques pour calculer des solutions et non à des fins de démonstration.

Pour un système d'ordre n nous avons $3 \left(\frac{n!}{2} \cdot (n+1) \right) + n$ opérations en virgules flottantes à réaliser. Pour un système 10×10 cela nous donne 59 875 210 opérations (de quoi donner le tourni !).

4.4 Elimination de gauss-Jordan par pivotement partiel.



Carl Friedrich Gauss (1777-1855) mathématicien, astronome et physicien Allemand



Marie Ennemond Camille Jordan (1838-1922) mathématicien Français

Nous allons ici décrire une autre méthode de résolution du système $A \times X = B$.

La méthode initiale dite d'élimination directe de Gauss souffre de quelques lacunes :

- Ne permet pas de résoudre un système matriciel dont un des termes diagonaux est nul.
- Possède une instabilité numérique due aux erreurs d'arrondis de calcul lorsque les termes de la matrice ne sont pas du même ordre de grandeur.

Cette méthode a peu à peu été améliorée en intégrant la possibilité d'inverser des lignes (et/ou colonnes) puis en choisissant judicieusement un 'pivot' qui permet de contraindre la matrice et d'éviter les divergences numériques.

Une méthode à pivotement complet permet de choisir un pivot dans toute la sous matrice $l \times l$ de la ligne l traitée. Elle nécessite de garder en mémoire l'historique de toutes les manipulations qui modifient l'ordre des solutions dans le vecteur solution.

La méthode à pivotement partiel qui sera exposée ci-après limite le choix du pivot aux termes sous le terme diagonal (même colonne), et ne change pas l'ordre du vecteur solution.

Le choix du *meilleur pivot* n'a jusqu'alors pas eu de solution exacte mais l'expérience a montré que le critère de choix du maximum (en valeur absolue) donnait de bons résultats.

A contrario de la méthode des déterminants, cette méthode nécessite beaucoup moins de temps de calcul et de stockage mémoire.

La matrice initiale est transformée (détruite) et le système est remplacé par un système équivalent dont le déterminant est différent de celui du système initial (en valeur absolue due à la normalisation des lignes et en signe due à l'inversion des lignes).

4.4.1 Effectuons le calcul sur un exemple :

$$\begin{bmatrix} 3 & -2 & 0 \\ 1 & -1 & -1 \\ 4 & -2 & -2 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 7 \\ 6 \end{bmatrix}$$

L'idée est que par transformation de ce système par combinaisons linéaires des lignes on puisse obtenir un système

trigonal supérieur de la forme $\begin{bmatrix} 1 & m_{1,2} & m_{1,3} \\ 0 & 1 & m_{2,3} \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$ limitant au maximum les erreurs d'arrondis fait dans

les calculs. Ensuite par résolution inverse (en résolvant les inconnues de la dernière à la première) on détermine ces inconnues. En effet on remarque que la dernière équation fournit alors dans l'ordre $z = \frac{c_3}{1} = c_3$, $y = c_2 - m_{2,3} \cdot z$ puis $x = c_1 - m_{1,2} \cdot y - m_{1,3} \cdot z$ en final.

Recherche du pivot Dans la première colonne, c'est-à-dire de l'élément ayant la plus grande valeur absolue. Ici il

s'agit du de l'élément de la troisième ligne :

$$\rightarrow \begin{bmatrix} 3 & -2 & 0 \\ 1 & -1 & -1 \\ 4 & -2 & -2 \end{bmatrix}$$

Inversion de la ligne pivot avec la ligne 1 $L_3 \leftrightarrow L_1$:

$$\begin{bmatrix} 4 & -2 & -2 \\ 1 & -1 & -1 \\ 3 & -2 & 0 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 6 \\ 7 \\ 2 \end{bmatrix}.$$

Division de la ligne 1 par le pivot $piv1 = 4$ (mise à 1 de la valeur diagonale)

$$\begin{bmatrix} \frac{4}{piv1} & \frac{-2}{piv1} & \frac{-2}{piv1} \\ 1 & -1 & -1 \\ 3 & -2 & 0 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{6}{piv1} \\ 7 \\ 2 \end{bmatrix}$$

ce qui produit

$$\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 1 & -1 & -1 \\ 3 & -2 & 0 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{3}{2} \\ 7 \\ 2 \end{bmatrix}.$$

Annulation progressive en remontant des termes $m_{3,1}$ et $m_{3,2}$ par combinaisons linéaires des lignes

$$\begin{cases} L_3 = L_3 - m_{3,1} \times L_1 \\ L_2 = L_2 - m_{2,1} \times L_1 \end{cases} \text{ ce qui produit } \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 1-1 & -1+\frac{1}{2} & -1+\frac{1}{2} \\ 3-3 & -2+3 \cdot \frac{1}{2} & 3 \cdot \frac{1}{2} \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{3}{2} \\ 7-\frac{3}{2} \\ 2-3 \cdot \frac{3}{2} \end{bmatrix}$$

soit

$$\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{3}{2} \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{3}{2} \\ \frac{11}{2} \\ -\frac{5}{2} \end{bmatrix}.$$

Nous allons recommencer la manipulation pour la ligne 2 :

Choisir le pivot entre $m_{2,2}$ et $m_{3,2}$ ici $piv2 = -\frac{1}{2}$.

Cette fois il est inutile d'inverser les lignes. Il suffit donc d'effectuer la mise à 1 du terme diagonal $m_{2,2}$ par division

de la ligne 2 par $piv2$

$$\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & 1 & 1 \\ 0 & -\frac{1}{2} & \frac{3}{2} \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{3}{2} \\ -11 \\ -\frac{5}{2} \end{bmatrix}$$

Elimination du terme $m_{3,2}$ par combinaison linéaire $L_3 = L_3 + \frac{1}{2}L_2$

$$\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{3}{2} \\ -11 \\ -8 \end{bmatrix}$$

Normalisation de la dernière ligne $piv3 = 2$ par division de la ligne par $piv3$

$$\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{3}{2} \\ -11 \\ -4 \end{bmatrix}$$

Ce qui donne les solutions :

$$\begin{cases} x = -4 \\ y = -7 \\ z = -4 \end{cases}$$

En prime, nous trouvons le déterminant de la matrice initiale

$$\det = piv1 \cdot piv2 \cdot piv3 \cdot (-1)^{nb \text{ inversion lignes}} = 4 \cdot (-1)^1 = -4$$

4.4.2 Généralisation et algorithme

Nous allons supposer ici résoudre le système $M \times X = B$ $M = [m_{i,j}]_{n,n}$; $X = [x_i]_n$; $B = [b_i]_n$

L'opération se fera en deux temps : la trigonalisation supérieure du système puis la détermination des inconnues.

Pour la trigonalisation, nous pouvons l'effectuer avec quatre raffinements successifs :

- Sans pivotement : Nous ne faisons aucunes inversions de lignes et supposons à chaque instant que le coefficient diagonal de la ligne traitée est non nul. Il s'agit là de la limitation de la méthode d'élimination de Gauss originale qui provoque une division par zéro dans ce cas particulier (qui est relativement courant cependant).
- Première amélioration pour pallier au défaut de division par zéro possible : En cas de nullité du coefficient $m_{l,l}$, on effectue une inversion de ligne avec la ligne tel que $m_{i,l} \neq 0$ pour $i > l$. On effectue alors l'inversion des lignes l et i .
- Avec pivotement partiel : On recherche sur les lignes en dessous de la ligne diagonale d'indice l la ligne dont le coefficient $m_{l,i}$ est maximal en valeur absolue. On effectue alors l'inversion des lignes l et i .
- Avec pivotement total : On recherche dans la sous-matrice $(l \times l)$ le coefficient $m_{j,i}$ maximal en valeur absolue. On effectue alors l'inversion des lignes l et i ainsi que l'inversion des colonnes l et j . Cette dernière inversion provoque aussi une inversion de l'ordre des solutions x_l et x_j . Il faudra donc garder en mémoire l'historique des inversions pour pouvoir résoudre le système au final.

Dés que l'on inverse des lignes et des colonnes, il faut faire attention au changement de signes successifs lors du calcul du déterminant à partir des pivots. Le déterminant global de la matrice peut se calculer par

$$\det = piv1 \cdot piv2 \cdot piv3 \cdot (-1)^{(nb \text{ inversion lignes} + nb \text{ inversion colonnes})}$$

Il s'avère que la dernière méthode n'améliore que peu le résultat en termes de précision. Par contre elle amène de la complexité algorithmique, du temps calcul et du stockage mémoire supplémentaire. Dans la plupart des utilisations de la méthode Gauss-Jordan, on se limitera au pivotement partiel.

4.4.2.1 La trigonalisation

Algorithmique	Exemple de programme C++
	<pre>real triangleGaussJordan(real* M, real * B, int size) { #define M(x,y) M[(x)*size+(y)] #define swap(x,y) {real s ; s=(x) ; (x)=(y) ; (y)=s ;} real determinant = 1 ;</pre>

<pre> determinant=1 Pour l de 1 à n Faire maxi= m_{l,l} npivot=l Pour nl de l à n Faire Si m_{nl,l} >maxi Alors npivot=nl maxi= m_{nl,l} pivot=m_{l,npivot} Si pivot=0 Alors Retourner 0 Si l!=npivot Alors inverser lignes l et npivot inverser constantes b_l et b_{npivot} determinant=-determinant determinant=determinant*pivot m_{l,l}=1 Pour c de l+1 à n Faire m_{l,c}=m_{l,c}/pivot b_l=b_l/pivot Pour nl de l+1 à n Faire Pour nc de l+1 à n Faire m_{nl,nc}=m_{nl,nc}-m_{nl,l}*m_{l,nc} b_{nl}=b_{nl}-m_{nl,l}*b_l m_{nl,l}=0 Retourner déterminant </pre>	<pre> for (int l = 0 ; l < size ; l++) { real maxi = M(l,l) ; int npivot = l ; for (int nl = l + 1 ; nl < size ; nl++) if (abs(M(nl,l)) > maxi) { maxi = abs(M(nl,l)) ; npivot = nl ; } if (maxi < _Epsilon) return 0 ; real pivot = M(npivot,l) ; if (l != npivot) { // inversion lignes et constantes for (int k = l ; k < size ; k++) swap(M(l,k),M(npivot,k)) ; swap(B[l],B[npivot]) ; determinant *= -1 // changement de signe ; } determinant *= pivot ; // normalisation de la ligne courante M(l,l) = 1 ; for (int c = l+1 ; c < size ; c++) M(l,c) /= pivot ; B[l] /= pivot ; // elimination et modification lignes en dessous for (int nl = l ; nl < size ; nl++) { real coef = M(nl,l) ; for (int nc = l + 1 ; nc < size ; nc++) M(nl,nc) -= M(l,nc) * coef ; B[nl] -= B[l] * coef ; M(l,nl) = 0 ; } } return determinant ; #undef M #undef swap } </pre>
--	---

Nous remarquons au passage, que nous utilisons deux propriétés des déterminants concernant la multiplication d'une ligne par un coefficient (ici le pivot) ainsi que l'invariance en cas d'ajout de combinaisons linéaires de lignes.

4.4.2.2 Le calcul inverse des résultats

L'algorithme ci-dessous présuppose que l'étape de trigonalisation supérieure de la matrice M à été correctement effectuée.

Algorithmique	Exemple de programme C++
<pre> X_{taille}=B_{taille} Pour l de n-1 à 1 Faire X_l=B_l Pour nc de n à l+1 Faire X_l=X_l-m_{l,nc}*X_{nc} </pre>	<pre> void calculGaussJordantriangle(real* M, real * B, real* X ,int size) { #define M(x,y) M[(x)*size+(y)] X[size-1]=B[size-1] ; for (int l = size-2; l>=0 ; l--) { X[l]=B[l]; for (int nc = size-1 ; nl >=0 size; nc--) X[l]=M[l * size+nc]*X[nc]; } #undef M } </pre>

4.4.3 Efficacité de la méthode

En terme d'efficacité, la résolution d'un tel système ne demande pas de mémoire supplémentaire que celle de stockage de la matrice et des vecteurs initiaux, soit $n \cdot (n + 2)$ cases de stockage de nombres réels. Le nombre d'opérations flottantes nécessaires est de $2 \cdot (n \cdot (n - 1)^2) \cdot (2 \text{opérations})$ pour la trigonalisation, suivi de

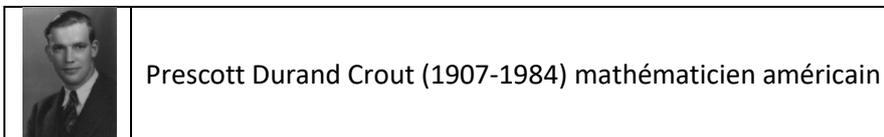
$1 + 2 \sum_{i=2}^{i=n} i = 1 + n \cdot (n-1)$ opérations pour la détermination des inconnues. Au total il y aura

$n \cdot (n-1) \cdot (4 \cdot (n-1) + 1)$ opérations flottantes.

Pour un système 10x10 nous obtenons approximativement 3330 opérations flottantes à comparer aux 60000000 nécessaires pour calculer l'inverse de la matrice par la méthode des déterminants ! En ce qui concerne la mémoire, nous nous cantonnons à 10x12 cases mémoire soit 120 à comparer à celles utiles pour calculer le déterminant soit $10^2 + 9^2 + \dots + 2^2 = 384$ cases. Une gigantesque économie qui permet de réaliser le calcul sur un processeur bas de gamme en un temps acceptable. Sur un 398p à 0.1Mflop, le temps de calcul est de $3330 / 0.1E6 = 0.033s$ ou 30 calculs par seconde ! en comparaison aux 10 minutes trouvées pour calculer le déterminant par la méthode classique !

Le revers de la médaille (car évidemment rien n'est noir et rien n'est blanc) est que cette méthode détruit la matrice initiale. Il suffit alors de la recopier avant d'effectuer le travail si on doit s'en resservir. Certes dans ce cas, le gain en mémoire initial, de 384/120 diminue à 384/240 mais il reste tout de même toujours supérieur à 1.

4.5 Décomposition L(lower)U(upper) par méthode de Crout.



Dans le cadre de la recherche d'efficacité de calcul en terme de vitesse et de mémoire, il existe une autre méthode de résolution de la désormais fameuse équation $A \times X = B$. Elle se base sur une propriété des matrices inversibles $A = L \times U$ avec L une matrice trigonale inférieure (low) et U une matrice trigonale supérieure (up). L'algorithme de calcul de ces deux matrices à été mis en place par un mathématicien presque contemporain (relativement par rapport aux autres) : M. Crout.

Cette décomposition (outre sa célérité un peu plus grande que la méthode de gauss) permet de pré-résoudre l'équation $A \times X = Y$ avec au moment du traitement Y inconnu. En effet cette décomposition (si on n'ajoute pas l'option pivotage partiel) ne touche en rien le deuxième membre de l'équation (ce que fait largement une résolution par la méthode de gauss et dérivées).

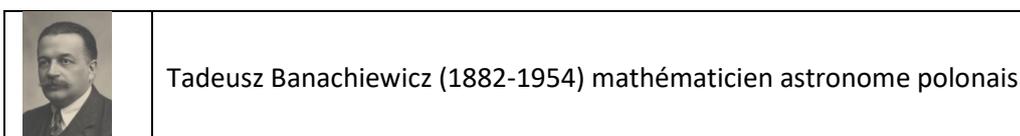
En fin de traitement, $A \times X = B$ soit $L \times U \times X = B$ se résout en deux fois

$$L \times U \times X = B \Leftrightarrow L \times (U \times X) = B \text{ donne } \begin{cases} L \times Y = B \\ U \times X = Y \end{cases} \text{ la « trigonalité » des matrices U et L rendent ces}$$

résolutions extrêmement rapides. Ces résolutions ne nécessitant le stockage intermédiaire que d'un vecteur de la taille du vecteur des constantes. De plus une astuce de stockage, permettra de conserver les deux matrices dans l'espace de stockage initial de la matrice A.

La matrice initiale est transformée (détruite) et le système est remplacé par un système équivalent dont le déterminant est différent de celui du système initial (en signe due à l'inversion des lignes).

Nota : La décomposition LU a été introduite par le mathématicien Tadeusz Banachiewicz en 1938.



Deux algorithmes découlent de cette décomposition connus sous les noms de méthode de Crout et méthode de Doolittle.



Myrick Haskell Doolittle (1830-1913) mathématicien américain

En fait, les deux méthodes diffèrent par le positionnement des 1 arbitraires sur les diagonales des matrices L ou U. Dans la méthode de Crout, la matrice triangulaire supérieure reçoit les 1 arbitraires dans sa diagonale tandis que dans la méthode de Doolittle c'est la matrice triangulaire inférieure qui reçoit les 1 arbitraires sur sa diagonale. Il en résulte une différence d'ordre de traitement des équations pour déterminer les différents coefficients. En fin de compte les deux méthodes sont complètement équivalentes du point de vue de leur consommation de ressources informatiques.

4.5.1 Exemple introductif

Mais assez de verbiage passons au travaux pratiques !

Reprenons notre exemple :
$$\begin{bmatrix} 3 & -2 & 0 \\ 1 & -1 & -1 \\ 4 & -2 & -2 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 7 \\ 6 \end{bmatrix} .$$

Nous pouvons montrer que
$$A = \begin{bmatrix} 3 & -2 & 0 \\ 1 & -1 & -1 \\ 4 & -2 & -2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1/3 & 1 & 0 \\ 4/3 & -2 & 1 \end{bmatrix} \times \begin{bmatrix} 3 & -2 & 0 \\ 0 & -1/3 & -1 \\ 0 & 0 & -4 \end{bmatrix} = L \times U$$

La détermination de L et U passe par la résolution de n^2 équations du style $a_{i,j} = \sum_{k=1}^{k=n} l_{i,k} \cdot U_{k,j}$. En imposant les coefficients d'une diagonale (de U ou L) égaux à 1, nous avons bien n^2 coefficients à déterminer (9 dans l'exemple). Le mérite de M. Crout aura été d'organiser ces équations en remarquant qu'en s'y prenant bien, la résolution se fait de proche en proche allant de 1 à n pour $u_{l,k}$ suivi des $l_{k,l}$ k variant de l à n. Pour faire simple, on dira que l'algorithme est trivial (encore fallait-il le trouver !).

La résolution finale de l'équation initiale s'effectue en deux temps : détermination de $Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$ tel que

$$L \times Y = \begin{bmatrix} 2 \\ 7 \\ 6 \end{bmatrix} \text{ soit } Y = \begin{bmatrix} 2 \\ 19/3 \\ -16/3 \end{bmatrix} \text{ s'ensuit : } U \times X = \begin{bmatrix} 2 \\ 19/3 \\ -16/3 \end{bmatrix} \text{ soit } X = \begin{bmatrix} 2 \\ 7 \\ 6 \end{bmatrix}$$

4.5.2 Généralisation

4.5.2.1 En restant simple :

Pour vous éviter une consommation trop grande d'aspirine, on retiendra juste que :

- Toute matrice inversible A peut s'écrire sous la forme $A = L \times U$. L et U deux matrices trigonales (L triangulaire inférieure et U triangulaire supérieure). Cette égalité matricielle conduit à n^2 équations scalaires.
- Deux possibilités : L matrice triangulaire inférieure avec des 1 sur la diagonale et U matrice triangulaire supérieure ou, L matrice triangulaire inférieure et U matrice triangulaire supérieure avec des 1 sur la diagonale. Les n^2

équations scalaires permettent donc de déterminer les inconnues de L : $\frac{n^2 - n}{2}$ inconnues, de U $\frac{n^2 - n}{2} + n$

inconnues, soit au total $\frac{n^2 - n}{2} + \frac{n^2 - n}{2} + n = n^2$ inconnues.

- Avec une telle décomposition, il est possible de résoudre aisément $A \times X = B \Leftrightarrow (L \times U) \times X = B$ avec une succession de constantes B_i inconnues au moment de la décomposition LU , sans pour autant toucher à cette décomposition lors de la résolution finale de $A \times X_i = B_i$.

4.5.2.2 Pour les adeptes de la migraine

$L \times U = A$ pour une matrice 4x4 se traduit par :

$$\begin{bmatrix} 1 = l_{1,1} & 0 & 0 & 0 \\ l_{2,1} & 1 = l_{2,2} & 0 & 0 \\ l_{3,1} & l_{3,2} & 1 = l_{3,3} & 0 \\ l_{4,1} & l_{4,2} & l_{4,3,1} & 1 = l_{4,4} \end{bmatrix} \times \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ 0 & u_{2,2} & u_{2,3} & u_{2,4} \\ 0 & 0 & u_{3,3} & u_{3,4} \\ 0 & 0 & 0 & u_{4,4} \end{bmatrix} = \begin{bmatrix} l_{1,1}u_{1,1} & l_{1,1}u_{1,2} & l_{1,1}u_{1,3} & l_{1,1}u_{1,4} \\ l_{2,1}u_{1,1} & l_{2,1}u_{1,2} + l_{2,2}u_{2,2} & l_{2,1}u_{1,3} + l_{2,2}u_{2,3} & l_{2,1}u_{1,4} + l_{2,2}u_{2,4} \\ l_{3,1}u_{1,1} & l_{3,1}u_{1,2} + l_{3,2}u_{2,2} & l_{3,1}u_{1,3} + l_{3,2}u_{2,3} + l_{3,3}u_{3,3} & l_{3,1}u_{1,4} + l_{3,2}u_{2,4} + l_{3,3}u_{3,4} \\ l_{4,1}u_{1,1} & l_{4,1}u_{1,2} + l_{4,2}u_{2,2} & l_{4,1}u_{1,3} + l_{4,2}u_{2,3} + l_{4,3}u_{3,3} & l_{4,1}u_{1,4} + l_{4,2}u_{2,4} + l_{4,3}u_{3,4} + l_{4,4}u_{4,4} \end{bmatrix} = [a_{i,j}]$$

En numérotant les équations et en les résolvant dans l'ordre (ordre de Krout), nous nous apercevons que dans chaque équation il ne reste qu'une inconnue (en vert foncé). En y regardant de plus près, on détecte une méthode de résolution. On remarque entre autre que les $a_{i,j}$ n'apparaissent toujours qu'une fois dans l'équation permettant de calculer le terme $l_{i,j}$ ou $u_{i,j}$. Nous pouvons donc replacer après calcul le terme $l_{i,j}$ ou $u_{i,j}$ dans la matrice initiale à la place du $a_{i,j}$ initial. Cette méthode évite de consommer de la mémoire.

$$\begin{cases} 1: l_{1,1}u_{1,1} = a_{1,1} \\ 2: l_{1,1}u_{1,2} = a_{1,2} \\ 3: l_{1,1}u_{1,3} = a_{1,3} \\ 4: l_{1,1}u_{1,4} = a_{1,4} \end{cases} \Rightarrow \begin{cases} 5: l_{2,1}u_{1,1} = a_{2,1} \\ 6: l_{3,1}u_{1,1} = a_{3,1} \\ 7: l_{4,1}u_{1,1} = a_{4,1} \end{cases} \Rightarrow \begin{cases} 1..4 \text{ permet de calculer } u_{1,i} \ i \in \{1, 2, 3, 4\} \\ 5..7 \text{ permet de calculer } l_{i,1} \ i \in \{2, 3, 4\} \end{cases}$$

$$\begin{cases} 8: l_{2,1}u_{1,2} + l_{2,2}u_{2,2} = a_{2,2} \\ 9: l_{2,1}u_{1,3} + l_{2,2}u_{2,3} = a_{2,3} \\ 10: l_{2,1}u_{1,4} + l_{2,2}u_{2,4} = a_{2,4} \end{cases} \Rightarrow \begin{cases} 11: l_{3,1}u_{1,2} + l_{3,2}u_{2,2} = a_{3,2} \\ 12: l_{4,1}u_{1,2} + l_{4,2}u_{2,2} = a_{4,2} \end{cases} \Rightarrow \begin{cases} 8..10 \text{ permet de calculer } u_{2,i} \ i \in \{2, 3, 4\} \\ 11..12 \text{ permet de calculer } l_{i,2} \ i \in \{3, 4\} \end{cases}$$

$$\begin{cases} 13: l_{3,1}u_{1,3} + l_{3,2}u_{2,3} + l_{3,3}u_{3,3} = a_{3,3} \\ 14: l_{3,1}u_{1,4} + l_{3,2}u_{2,4} + l_{3,3}u_{3,4} = a_{3,4} \end{cases} \Rightarrow \begin{cases} 15: l_{4,1}u_{1,3} + l_{4,2}u_{2,3} + l_{4,3}u_{3,3} = a_{4,3} \\ 15 \text{ permet de calculer } l_{i,3} \ i \in \{4\} \end{cases}$$

$$16: l_{4,1}u_{1,4} + l_{4,2}u_{2,4} + l_{4,3}u_{3,4} + l_{4,4}u_{4,4} = a_{4,4} \Rightarrow (16 \text{ permet de calculer } u_{2,i} \ i \in \{3, 4\})$$

Voici un exemple de programmes des deux décompositions sans pivotement écrites en C++. On adaptera aisément dans d'autres langages :

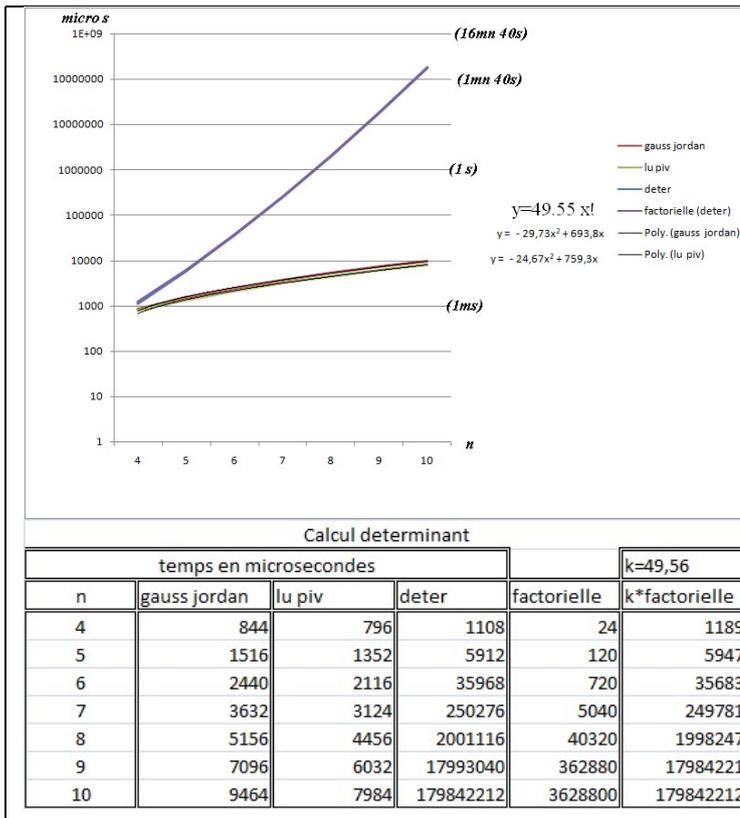
Programme C++ de décomposition Doolittle	Programme C++ de décomposition de CROUT
<pre> boolean luDecompDoolittle(real *M, int size){ #define M(i, j) M[(i)*size + j] int i, j, k, p; real s; for (k = 0; k < n; k++) { for (j = k; j < n; j++) { s=0.0; for (p = 0; p < k; p++) s=M(k,p)* M(p,j); M(k,j) -=s; } If (abs(M(k,k))<_Epsilon) return false ; for (i = k+1 i < n; i++) { s=0.0 for (p = 0 p < k; p++) s = M(i,p) * M(p,k); M(i,k)-=s; M(i,k)/= M(k,k); } } return true ; #undef M } </pre>	<pre> boolean luDecompCrout(real* M, int size) { #define M(i, j) M[(i)*size + j] int i, j, k; real s; real coef; for (k = 0; k < size; k++) { for (i = k; i < size; i++) { s = 0.0; for (j = 0; j < k; j++) s += M(i, j) * M(j, k); M(i, k) -= s; } If (abs(M(k,k))<_Epsilon) return false ; coef = 1.0 / M(k, k); for (j = k + 1; j < size; j++) { s = 0.0; for (i = 0; i < k; i++) s += M(k, i) * M(i, j); M(k, j) = (M(k, j) - s) * coef; } } return true ; #undef M } </pre>

$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{2,1} & 1 & 0 & 0 \\ l_{3,1} & l_{3,2} & 1 & 0 \\ l_{4,1} & l_{4,2} & l_{4,3} & 1 \end{bmatrix} U = \begin{bmatrix} u_{1,1} & u_{1,1} & u_{1,1} & u_{1,1} \\ 0 & u_{1,1} & u_{1,1} & u_{1,1} \\ 0 & 0 & u_{1,1} & u_{1,1} \\ 0 & 0 & 0 & u_{1,1} \end{bmatrix}$ $M_{\text{apres calcul}} = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ l_{2,1} & u_{2,2} & u_{2,3} & u_{2,4} \\ l_{3,1} & l_{3,2} & u_{3,3} & u_{3,4} \\ l_{4,1} & l_{4,2} & l_{4,3} & u_{4,4} \end{bmatrix} = [m_{i,j}]$ <p> si($i < j$) $l_{i,j} = 0$ sinon {si($i = j$) $l_{i,j} = 1$ sinon $l_{i,j} = m_{i,j}$} si($i \leq j$) $u_{i,j} = m_{i,j}$ sinon $u_{i,j} = 0$ </p>	$L = \begin{bmatrix} l_{1,1} & 0 & 0 & 0 \\ l_{2,1} & l_{2,2} & 0 & 0 \\ l_{3,1} & l_{3,2} & l_{3,3} & 0 \\ l_{4,1} & l_{4,2} & l_{4,3} & l_{4,4} \end{bmatrix} U = \begin{bmatrix} 1 & u_{1,1} & u_{1,1} & u_{1,1} \\ 0 & 1 & u_{1,1} & u_{1,1} \\ 0 & 0 & 1 & u_{1,1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$ $M_{\text{apres calcul}} = \begin{bmatrix} l_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ l_{2,1} & l_{2,2} & u_{2,3} & u_{2,4} \\ l_{3,1} & l_{3,2} & l_{3,3} & u_{3,4} \\ l_{4,1} & l_{4,2} & l_{4,3} & l_{4,4} \end{bmatrix} = [m_{i,j}]$ <p> si($i \geq j$) $l_{i,j} = m_{i,j}$ sinon $l_{i,j} = 0$ si($i < j$) $u_{i,j} = 0$ sinon {si($i = j$) $u_{i,j} = 1$ sinon $u_{i,j} = m_{i,j}$} </p>
--	--

4.6 Comparaison des méthodes

Un jeu de test a été effectué sur un microcontrôleur de type ATMega2560 (processeur à 16Mhz) pour comparer le calcul du déterminant par les trois méthodes exposées ci-dessus. Nous avons choisi ce processeur pour sa taille mémoire ram de 8ko et pour sa vitesse permettant de juger en microsecondes l'efficacité des méthodes.

Le graphique ci-dessous donne les résultats sous forme de courbes du temps en $\mu s = 10^{-6} s$ en fonction de la taille de la matrice traitée $n \times n$:



Pour chaque courbe nous avons calculé (ou fait calculer par Excel lorsque cela était possible), une courbe de tendance (un lissage polynomial !).

Les résultats sont sans appel :

- Les méthodes de Gauss et Crout sont de l'ordre de n^3 en terme de nombre d'opérations.
- On note un léger avantage pour la méthode LU, mais cet avantage se réduit lorsqu'on calcule une solution à un système. En effet la méthode LU nécessite deux fois plus d'opérations dans sa résolution finale que la méthode de Gauss.
- En ce qui concerne la méthode des déterminants (sans parler de la résolution qui nécessite de calculer $n + 1$ déterminants) elle est de l'ordre de $n! = n \cdot (n - 1) \cdot (n - 2) \cdot \dots \cdot 2$

En un mot comme en dix, la méthode des déterminants est catastrophique du point de vue de l'efficacité !

V Utilisation des matrices en géométrie vectorielle

Un vecteur peut être représenté par une matrice colonne (ou ligne), alors il n'y a rien de plus naturel que de « bricoler » la représentation de ces vecteurs à l'aide de l'outil matriciel ! En effet, il est possible, élégant et même pratique, de définir le résultat numérique des opérations géométriques de base (symétrie, rotation, homothétie ...) sous forme de produit matriciel. En mécanique théorique, le changement de repère se traduira lui aussi par un produit matriciel (mais n'est il pas lui-même une opération de rotation de la chaise de l'observateur du système ?). Dans le jargon des mathématiques, on appelle ces transformations des isométries (qui conserve (iso) les distances (metries)!) ou plus généralement les similitudes (qui transforment une forme en une forme similaire).

5.1 Exemple introductif

Je reprendrais ici notre exemple introductif au §1.1.2 (page 4).

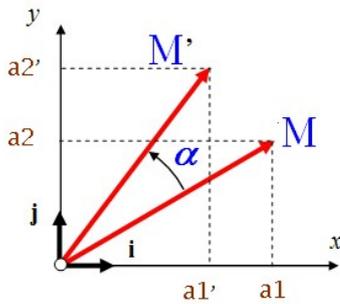
Pour la suite de ce chapitre nous nous positionnerons dans \mathbb{R}^3 , l'espace à trois dimensions dans lequel nous évoluons tous les jours sans le savoir !

5.2 Rotation autour d'un axe

Pour simplifier un peu l'écriture, dans ce qui suit nous adopterons les abréviations $\begin{cases} C\alpha = \cos(\alpha) \\ S\alpha = \sin(\alpha) \end{cases}$

soit $\overrightarrow{OM} = \begin{bmatrix} a1 \\ a2 \\ a3 \end{bmatrix}_{R_i}$ avec $R_i = \{ \vec{x}_i, \vec{y}_i, \vec{z}_i \}$ le repère vectoriel N°i

5.2.1 Axe \vec{z}



Une rotation d'angle α autour de \vec{z} peut se traduire par l'opération sur les composantes de \overline{OM} par

$$\overline{OM}' = R_{z,\alpha} \times \overline{OM}$$

$$\begin{bmatrix} a'1 \\ a'2 \\ a'3 \end{bmatrix}_{R_i} = \begin{bmatrix} C\alpha & -S\alpha & 0 \\ S\alpha & C\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} a1 \\ a2 \\ a3 \end{bmatrix}_{R_i} \quad \begin{cases} a'1 = C\alpha \cdot a1 - S\alpha \cdot a2 \\ a'2 = S\alpha \cdot a1 + C\alpha \cdot a2 \\ a'3 = a3 \end{cases}$$

5.2.2 Axe \vec{x}

soit $\overline{OM} = \begin{bmatrix} a1 \\ a2 \\ a3 \end{bmatrix}_{R_i}$ avec $R_i = \{\vec{x}_i, \vec{y}_i, \vec{z}_i\}$ le repère vectoriel N°i

Une rotation d'angle α autour de \vec{x} peut se traduire par l'opération sur les composantes de \overline{OM} par

$$\overline{OM}' = R_{x,\alpha} \times \overline{OM}$$

$$\begin{bmatrix} a'1 \\ a'2 \\ a'3 \end{bmatrix}_{R_i} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\alpha & -S\alpha \\ 0 & S\alpha & C\alpha \end{bmatrix} \times \begin{bmatrix} a1 \\ a2 \\ a3 \end{bmatrix}_{R_i} \quad \begin{cases} a'1 = a1 \\ a'2 = C\alpha \cdot a2 - S\alpha \cdot a3 \\ a'3 = S\alpha \cdot a2 + C\alpha \cdot a3 \end{cases}$$

5.2.3 Axe \vec{y}

Une rotation d'angle α autour de \vec{y} peut se traduire par l'opération sur les composantes de \overline{OM} par

$$\overline{OM}' = R_{y,\alpha} \times \overline{OM}$$

$$\begin{bmatrix} a'1 \\ a'2 \\ a'3 \end{bmatrix}_{R_i} = \begin{bmatrix} C\alpha & 0 & S\alpha \\ 0 & 1 & 0 \\ -S\alpha & 0 & C\alpha \end{bmatrix} \times \begin{bmatrix} a1 \\ a2 \\ a3 \end{bmatrix}_{R_i} \quad \begin{cases} a'1 = C\alpha \cdot a1 + S\alpha \cdot a3 \\ a'2 = a2 \\ a'3 = S\alpha \cdot a1 - C\alpha \cdot a3 \end{cases}$$

5.2.4 Axe quelconque \vec{u}

La démonstration de cette dernière transformation étant un tant soit peu gratte neurones, nous donnerons ici le résultat sans plus d'explications.

$$\vec{u} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}_{R_i} \text{ avec } u_x^2 + u_y^2 + u_z^2 = 1 \text{ (vecteur normé)}. \text{ On appelle les } u_n \text{ les cosinus directeurs de } \vec{u}.$$

Une rotation d'angle α autour de \vec{u} peut se traduire par l'opération sur les composantes de \overrightarrow{OM} par

$$\overrightarrow{OM'} = R_{u,\alpha}^- \times \overrightarrow{OM}$$

$$\begin{bmatrix} a'1 \\ a'2 \\ a'3 \end{bmatrix}_{R_i} = \begin{bmatrix} u_x^2(1-C\alpha) + C\alpha & u_x u_y(1-C\alpha) - u_z S\alpha & u_x u_z(1-C\alpha) + u_y S\alpha \\ u_x u_y(1-C\alpha) + u_z S\alpha & u_y^2(1-C\alpha) + C\alpha & u_y u_z(1-C\alpha) - u_x S\alpha \\ u_x u_z(1-C\alpha) - u_y S\alpha & u_y u_z(1-C\alpha) + u_x S\alpha & u_z^2(1-C\alpha) + C\alpha \end{bmatrix} \times \begin{bmatrix} a1 \\ a2 \\ a3 \end{bmatrix}_{R_i}$$

Remarque :

On retrouve dans la matrice de rotation $R_{u,\alpha}^-$ en colonnes successivement les composantes des trois vecteurs de la base initiale une fois 'tournés'.

$$\begin{cases} \vec{x}' = R_{u,\alpha}^- \times \vec{x}_i = \begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix}_{R_i} \\ \vec{y}' = R_{u,\alpha}^- \times \vec{y}_i = \begin{pmatrix} y'_1 \\ y'_2 \\ y'_3 \end{pmatrix}_{R_i} \\ \vec{z}' = R_{u,\alpha}^- \times \vec{z}_i = \begin{pmatrix} z'_1 \\ z'_2 \\ z'_3 \end{pmatrix}_{R_i} \end{cases} \text{ avec } R_{u,\alpha}^- = \begin{bmatrix} x'_1 & y'_1 & z'_1 \\ x'_2 & y'_2 & z'_2 \\ x'_3 & y'_3 & z'_3 \end{bmatrix}$$

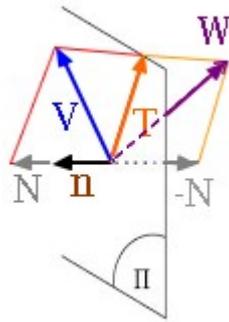
$$\text{On s'en persuadera en regardant } R_{z,\alpha}^- = \begin{bmatrix} C\alpha & -S\alpha & 0 \\ S\alpha & C\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5.3 Symétries planes

Dans une symétrie par rapport à un plan, de normale $\vec{n} = (n_x, n_y, n_z)_{R_i}$ nous pouvons décomposer le vecteur à

$$\text{symétriser } \vec{V} = \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix}_{R_i} \text{ en deux vecteurs, l'un normal au plan de symétrie } \vec{n} : \vec{N} = \begin{pmatrix} N_x \\ N_y \\ N_z \end{pmatrix}_{R_i} \text{ et l'autre dans le plan de}$$

symétrie : $\vec{T} = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}_{R_i}$ tel que $\begin{cases} \vec{V} = \vec{N} + \vec{T} \\ \vec{N} \wedge \vec{n} = \vec{0} \\ \vec{T} \cdot \vec{n} = 0 \end{cases}$



La solution est $N_x = n_x(n_x V_x + n_y V_y + n_z V_z)$ et $T_x = V_x - n_x(n_x V_x + n_y V_y + n_z V_z)$ les termes N_y, N_z, T_y, T_z sont obtenus par permutation circulaire des indices x, y et z. Nous trouvons donc le symétrique $\vec{W} = \vec{T} - \vec{N}$ et

$$\begin{cases} W_x = (1 - 2n_x^2)V_x - 2n_x n_y V_y - 2n_x n_z V_z \\ W_y = -2n_y n_x V_x + (1 - 2n_y^2)V_y - 2n_y n_z V_z \\ W_z = -2n_z n_x V_x - 2n_z n_y V_y + (1 - 2n_z^2)V_z \end{cases}$$

5.3.1 Par rapport au plan \vec{x}, \vec{y} de normale \vec{z}

$n_x = n_y = 0; n_z = 1$ La matrice de transformation M tel que $\vec{W} = M \times \vec{V}$ est $M_z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$ Nous aurions pu

nous éviter les calculs en disant que seul la composante sur z changeait de signe.

5.3.2 Par rapport au plan \vec{y}, \vec{z} de normale \vec{x}

$n_y = n_z = 0; n_x = 1$ La matrice de transformation M tel que $\vec{W} = M \times \vec{V}$ est $M_x = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

5.3.3 Par rapport au plan \vec{z}, \vec{x} de normale \vec{y}

$n_x = n_z = 0; n_y = 1$ La matrice de transformation M tel que $\vec{W} = M \times \vec{V}$ est $M_y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

5.3.4 Par rapport à un plan de normale \vec{n}

La matrice de transformation M tel que $\vec{W} = M \times \vec{V}$ est $M_n = \begin{bmatrix} 1 - 2n_x^2 & -2n_x n_y & -2n_x n_z \\ -2n_y n_x & 1 - 2n_y^2 & -2n_y n_z \\ -2n_z n_x & -2n_z n_y & 1 - 2n_z^2 \end{bmatrix}$ avec

$$n_x^2 + n_y^2 + n_z^2 = 1.$$

5.4 Homothéties

Nous quittons ici le monde restreint des isométries pour se placer dans un « hyper espace » correspondant celui des similitudes.

Les opérations d'homothétie servent dans la « vraie vie » à remettre à l'échelle un objet (virtuel) en CAO, dans tous les processus de visualisation numérique (effectuer un zoom ...) On peut en distinguer deux types : le grandissement global et le grandissement suivant un axe (on pensera aux effets spéciaux dans les visiteurs quand le

nez de *Jacquouille* se déforme !!).

5.4.1 Globale

C'est le cas le plus simple $\vec{W} = k \cdot \vec{V}$ avec k le facteur d'homothétie.

$$M \text{ tel que } \vec{W} = k\vec{V} = M_k \times \vec{V} \text{ est } M_x = \begin{bmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & k \end{bmatrix}$$

5.4.2 Dans une direction \vec{n}

En reprenant notre raisonnement de la symétrie par rapport à un plan quelconque de normale \vec{n} du §5.3 (P 29) nous avons $\vec{W} = \vec{T} + k\vec{N}$ La matrice de transformation se déduit donc des équations du 5.3

$$\begin{cases} W_x = (1 + n_x^2(k-1))V_x + (k-1)n_x n_y V_y + (k-1)n_x n_z V_z \\ W_y = (k-1)n_y n_x V_x + (1 + n_y^2(k-1))V_y + (k-1)n_y n_z V_z \\ W_z = (k-1)n_z n_x V_x + (k-1)n_z n_y V_y + (1 + n_z^2(k-1))V_z \end{cases} \text{ soit la matrice de transformation}$$

$$M_{n,k} = \begin{bmatrix} 1 + n_x^2(k-1) & (k-1)n_x n_y & (k-1)n_x n_z \\ (k-1)n_y n_x & 1 + n_y^2(k-1) & (k-1)n_y n_z \\ (k-1)n_z n_x & (k-1)n_z n_y & 1 + n_z^2(k-1) \end{bmatrix}$$

On remarquera que la symétrie par rapport à un plan repéré par son vecteur normal, correspond à une homothétie de rapport -1 suivant ce même vecteur...

5.4.3 Projection sur un plan de normale \vec{n}

Projeter un vecteur sur un plan est en fait une homothétie par rapport à un axe dont le coefficient d'homothétie est de 0 La matrice de transformation se déduit donc de celle du 5.4.2 en posant k=0.

$$PROJ_n = \begin{bmatrix} 1 - n_x^2 & -n_x n_y & -n_x n_z \\ -n_y n_x & 1 - n_y^2 & -n_y n_z \\ -n_z n_x & -n_z n_y & 1 - n_z^2 \end{bmatrix} \text{ Une telle transformation est utile dans les processus de visualisation d'objets}$$

pour obtenir une image 'plate'.

5.5 Qu'en est-il des opérations inverses ?

Faire et défaire c'est toujours travailler ! Même en ce qui concerne les manipulations mathématiques. Nous avons souvent besoin d'effectuer une opération inverse d'une opération déjà effectuée sur un vecteur. Nous pouvons comme nous savons le faire désormais calculer la matrice inverse M^{-1} de la transformée M . C'est dans le cas présent « tuer une mouche avec un canon ». En effet de manière très simple nous trouvons les matrices inverses correspondant aux opérations sur vecteurs :

$$\text{Rotation : } (Mrot_{n,\alpha})^{-1} = Mrot_{n,-\alpha} \text{ on peut s'en convaincre en effectuant le produit } Mrot_{z,-\alpha} \times Mrot_{z,\alpha}$$

$$\begin{bmatrix} C\alpha & S\alpha \\ -S\alpha & C\alpha \end{bmatrix} \times \begin{bmatrix} C\alpha & -S\alpha \\ S\alpha & C\alpha \end{bmatrix} = \begin{bmatrix} C^2\alpha + S^2\alpha & -C\alpha S\alpha + S\alpha C\alpha \\ -S\alpha C\alpha + C\alpha S\alpha & S^2\alpha + C^2\alpha \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I_{2,2} \text{ (comme en politique, on retourne sa veste !)}$$

symétrie : $(M_{sym_n})^{-1} = M_{sym_n}$ (les ennemis de mes ennemis sont mes amis !)

homothétie : $(M_{homo_k})^{-1} = M_{homo_{1/k}}$ lorsque $k=1$ on retrouve l'opération nulle !

VI A propos de matrices homogènes

Non il ne s'agit pas ici de prendre une cuillère mathématique pour brasser les coefficients des matrices... Dans la partie V, nous avons vu comment réaliser des transformations vectorielles du type rotation, symétries et homothéties. Or lorsque l'on essaie de modéliser un système mécanique, on se retrouve confronté au problème des translations que l'on aimerait traiter aussi facilement que les transformations du V. Nous allons donc développer une « ruse » dans les outils de codage qui nous permettra d'inclure ces translations et certainement bien d'autres choses dans notre outil matriciel : les matrices homogènes.

6.1 Exemple introductif

Essayons de décrire le mouvement du poignet du robot (2D) ci-dessous :

Nous pouvons considérer que le mouvement du bras depuis sa position de repos $\lambda = 0; \alpha = 0$ s'étend de λ sur l'axe \vec{u} (initialement confondu avec l'axe \vec{x}), suivi d'une rotation de α de l'ensemble du bras autour de \vec{z} . Il y a donc deux transformations successives : translation puis rotation.

$\vec{OM} = \vec{OA} + \vec{AM} = d \cdot \vec{u} + \lambda \cdot \vec{u}$ ce qui donne dans le repère O, \vec{u}, \vec{v}

$\vec{OM} = \begin{bmatrix} d + \lambda \\ 0 \end{bmatrix}_{O, \vec{u}, \vec{v}}$ rajoutons artificiellement une coordonnée dite d'homogénéisation (1) au vecteur $\vec{OM}_{homogene} = \begin{bmatrix} d + \lambda \\ 0 \\ 1 \end{bmatrix}_{O, \vec{u}, \vec{v}}$ nous pouvons maintenant écrire matriciellement

$\vec{OM}_{homogene} = \begin{bmatrix} d + \lambda \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \lambda \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d \\ 0 \\ 1 \end{bmatrix}$ nous venons de créer la représentation matricielle d'une translation sur le premier axe de représentation de la matrice.

Pour décrire le mécanisme total nous pouvons écrire

$\vec{OM} = R_{z, \alpha} \cdot T_{x, \lambda} \cdot \vec{U}_{homo}$ dans lequel $\vec{U}_{homo} = \begin{bmatrix} d \\ 0 \\ 1 \end{bmatrix}$ représente la position initiale de M avant déplacements.

avec comme nous l'avons vu au §5.2.1 (P 27).

$$R_{z, \alpha} = \begin{bmatrix} C\alpha & -S\alpha \\ S\alpha & C\alpha \end{bmatrix} \text{ et donc } R_{z, \alpha}^{homo} = \begin{bmatrix} C\alpha & -S\alpha & 0 \\ S\alpha & C\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_{z,\alpha}^- \underset{\text{hom}o}{T}_{x,\lambda}^- = \begin{bmatrix} C\alpha & -S\alpha & 0 \\ S\alpha & C\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & \lambda \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C\alpha & -S\alpha & C\alpha \cdot \lambda \\ S\alpha & C\alpha & S\alpha \cdot \lambda \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Donc } R_{z,\alpha}^- \underset{\text{hom}o}{T}_{x,\lambda}^- \overrightarrow{U}_{\text{hom}o} = \begin{bmatrix} C\alpha & -S\alpha & C\alpha \cdot \lambda \\ S\alpha & C\alpha & S\alpha \cdot \lambda \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} d \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} C\alpha \cdot d + C\alpha \cdot \lambda \\ S\alpha \cdot d + S\alpha \cdot \lambda \\ 1 \end{bmatrix}$$

$$\text{soient les équations } \begin{cases} x_M = C\alpha \cdot (d + \lambda) \\ y_M = S\alpha \cdot (d + \lambda) \end{cases}$$

Ça y est vous avez compris, il suffit de rajouter une coordonnée qui n'en est pas une pour arriver dans le monde des « homogènes » (en fait cette coordonnée pourrait être autre chose que 1 et nous serions alors dans un monde parallèle mais la série "Star Trek" étant du domaine du passé, Spoke ne peut plus aller nous sauver dans ces mondes, nous en resterons donc au premier monde : le notre, le N°1 !). Il nous est maintenant possible de décrire une succession d'évolutions spatiales par une série de multiplications de matrices dans l'ordre des évolutions (et le bon s'il vous plaît !). Dans la suite nous indiquerons les matrices homogènes correspondant aux opérations du §V.

6.2 Translation homogène

$$\text{translation de vecteur } (t_x, t_y, t_z)_{\mathbb{R}^3} \underset{\text{hom}o}{T}_{(t_x, t_y, t_z)} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

6.3 Autres manipulations homogènes

Les matrices homogènes des autres manipulations se déduisent de celles évoquées au §V en :

- ajoutant une dimension (fictive),
- mettant le dernier terme en bas à droite à 1
- calant la matrice de la transformation décrite au §V dans le coin en haut à gauche de la matrice homogène.

$$\text{Trans}_{\text{hom}o} = \left[\begin{array}{ccc|c} & & & 0 \\ & \text{Trans} & & 0 \\ & & & 0 \\ \hline 0 & 0 & 0 & 1 \end{array} \right]$$

6.4 Domaines d'utilisation du système « homogène »

Comme vous l'avez compris dans l'exemple introductif, l'utilisation des matrices homogènes est très répandue dans les armoires de commande des robots. Cette dernière permet de donner la position finale du poignet (outil) du robot en fonction des positions des actionneurs qui motorisent les pivots et des éléments télescopiques qui constituent le bras du robot. L'opération inverse (détermination des positions des actionneurs en fonction de la position cartésienne souhaitée) est plus compliquée car lors de grands déplacements, les coefficients des matrices sont des fonctions circulaires (sin et cos) non linéaires. Pour de petits déplacements autour d'une position, on peut linéariser ces matrices (au premier ordre) et approximer par matrice inverse les positions des actionneurs. On retrouve ici la théorie des torseurs de petits déplacements utilisée en métrologie.

Le graphisme 3D utilise aussi beaucoup la technique des matrices homogènes pour déplacer et modifier des objets (représentés par un nuage de point donc de vecteurs) dans un espace cartésien.

Une utilisation de ces matrices dans le domaine du calcul de résistance des matériaux a fait l'objet d'un programme et d'un article du même auteur que celui du présent papier, et peuvent se télécharger sur le site altituduino.com ([http://altituduino.com/ media/matrices-de-transfert.pdf](http://altituduino.com/media/matrices-de-transfert.pdf)) ou en navigant sur le site dans « divers cours et article » et en allant sur « Les Matrices de Transfert Dans La Résolution Des Poutres Continues ».

6.5 soyons Fous !!

Soyons fous et allons plus loin : essayons d'unifier dans une même opération matricielle l'addition et la multiplication des nombres.

Remplaçons les nombres par un objet constitué d'un doublet homogène et des propriétés A (addition) et P

(produit) représentées par des matrices homogènes tel que $x \rightarrow x_h = \begin{bmatrix} x \\ 1 \end{bmatrix}; y \rightarrow y_h = \begin{bmatrix} y \\ 1 \end{bmatrix}; z \rightarrow z_h = \begin{bmatrix} z \\ 1 \end{bmatrix}$

$x.A = \begin{bmatrix} 1 & x \\ 0 & 1 \end{bmatrix}; x.P = \begin{bmatrix} x & 0 \\ 0 & 1 \end{bmatrix}$. Idem pour y et z. Nous voyons que le résultat homogène de $x \cdot y$ peut se calculer

par $x.P \times y_h = \begin{bmatrix} x & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} y \\ 1 \end{bmatrix} = \begin{bmatrix} x \cdot y \\ 1 \end{bmatrix}$ (lire propriété P de x appliquée à y) et que $x + y$ se calcule par

$x.A \times y_h = \begin{bmatrix} 1 & x \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} y \\ 1 \end{bmatrix} = \begin{bmatrix} y + x \\ 1 \end{bmatrix}$ (lire propriété A de x appliquée à y).

Étonnant non ?

Par la suite nous considérerons que tout calcul produit un nouvel objet auquel nous associons ses deux propriétés

.P et .A, stockées avec le vecteur homogène sous forme de matrices homogènes, soit : $(x + y).A = \begin{bmatrix} 1 & x + y \\ 0 & 1 \end{bmatrix}$ et

$(x + y).P = \begin{bmatrix} x + y & 0 \\ 0 & 1 \end{bmatrix}$.

Mais alors que devient le pendant homogène de $(x + y)z$? Et bien tout simplement

$(x.A \times y_h).P \times z_h = \left(\begin{bmatrix} 1 & x \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} y \\ 1 \end{bmatrix} \right) .P \times \begin{bmatrix} z \\ 1 \end{bmatrix} = \begin{bmatrix} x + y & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} z \\ 1 \end{bmatrix} = \begin{bmatrix} (x + y)z \\ 1 \end{bmatrix}$

À ce nouvel objet numérique, nous associons les propriétés .P et .A pour pouvoir faciliter les manipulations futures.

Pour les mathématiciens, on introduit l'élément neutre du produit $1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ avec $1.P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; 1.A = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$ ainsi

que l'élément absorbant du produit $0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ avec $0.P = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}; 0.A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ qui est aussi élément neutre de

l'addition. Cela a une odeur d'Evariste Galois, non ?

Toute cette « cuisine » sent fortement la programmation orientée objet des langages Ada C++ Java Ruby ... (et juste pour le citer car j'ai pour ce serpent informatique une aversion certaine : Python).

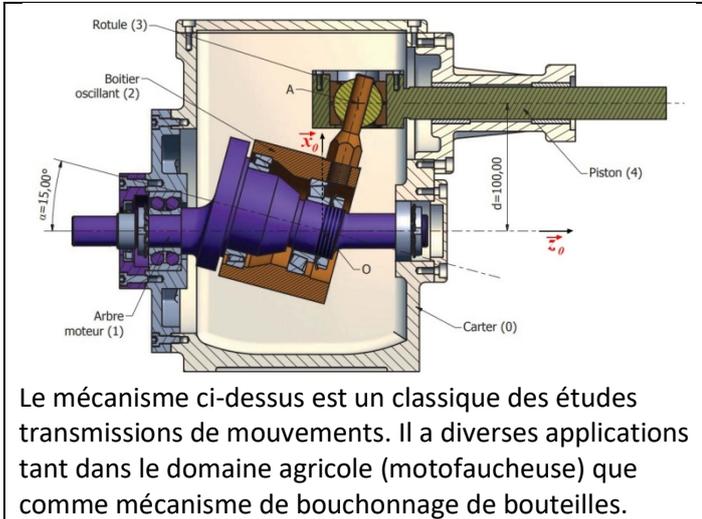
Cet exemple n'est certainement qu'une élucubration « fumeuse » mais elle montre une partie de l'étendue des possibilités de cette notion d'homogène (à ne pas mettre dans toutes les mains tout de même !).

VII Les changements de repères

Je reprends ici ma casquette de mécanicien (théorique). Les calculs de cinématique sur des mécanismes nécessitent entre autre d'écrire les équations de fermeture géométrique du mécanisme étudié. Pour ce faire, il est nécessaire

de décrire de manière la plus simple possible les éléments vectoriels de ces fermetures géométriques de boucles. Nous attachons à chaque pièce matérielle plusieurs repères qui se déduisent l'un de l'autre par des transformations les plus simples possibles (rotation autour d'un axe du repère initial). Il importe une fois la modélisation faite d'écrire les équations sous forme algébriques dans un même repère. Le mécanisme de passage d'un repère à un autre peut se faire « à la main » ou systématisé par l'intermédiaire de produits matriciels.

7.1 Exemple introductif



L'établissement des équations de cinématique issues de l'équation de fermeture géométrique nécessite la description de plusieurs repères directs attachés aux pièces :

- $\mathcal{R}_0 = (\vec{x}_0, \vec{y}_0, \vec{z}_0)$ attaché au bâti n°0
- $\mathcal{R}_{10} = (\vec{x}_{10}, \vec{y}_{10}, \vec{z}_{10})$ avec $\vec{z}_{10} = \vec{z}_0$ attaché à la manivelle n°1,-
- $\mathcal{R}_{11} = (\vec{x}_{11}, \vec{y}_{11}, \vec{z}_{11})$ avec $\vec{y}_{11} = \vec{y}_{10}$ attaché à la manivelle n°1,
- ...

Les repères successifs :

	$\begin{cases} \vec{x}_{10} = C\omega t \cdot \vec{x}_0 + S\omega t \cdot \vec{y}_0 \\ \vec{y}_{10} = -S\omega t \cdot \vec{x}_0 + C\omega t \cdot \vec{y}_0 \\ \vec{z}_{10} = \vec{z}_0 \end{cases}$ <p>(1) Pour éviter les erreurs, il est préférable de dessiner ses repères avec un petit angle positif (dans le sens du repère direct).</p>	<p>Pour $\vec{U} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}_{\mathcal{R}_0}$</p> $\vec{U} = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}_{\mathcal{R}_{10}} = \begin{bmatrix} C\omega t & S\omega t & 0 \\ -S\omega t & C\omega t & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}_{\mathcal{R}_0} = R_{z, \omega t} \begin{bmatrix} a \\ b \\ c \end{bmatrix}_{\mathcal{R}_0}$
--	--	---

	$\begin{cases} \vec{x}_{11} = C\beta \cdot \vec{x}_{10} - S\beta \cdot \vec{z}_{10} \\ \vec{y}_{11} = \vec{y}_{10} \\ \vec{z}_{11} = S\beta \cdot \vec{x}_{10} + C\beta \cdot \vec{z}_{10} \end{cases}$ <p>Ici on changé le paramètre du problème α pour se ramener à un paramètre algébriquement positif $\beta = -\alpha$ respectant (1).</p>	<p>Pour $\vec{U} = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}_{\mathcal{R}_{10}}$</p> $\vec{U} = \begin{bmatrix} a'' \\ b'' \\ c'' \end{bmatrix}_{\mathcal{R}_{11}} = \begin{bmatrix} C\beta & 0 & -S\beta \\ 0 & 1 & 0 \\ S\beta & 0 & C\beta \end{bmatrix} \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}_{\mathcal{R}_{10}} = R_{y, \beta} \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}_{\mathcal{R}_{10}}$
--	--	---

Par identification nous obtenons :

$$\begin{cases} \vec{x}_{11} = C\beta \cdot (C\omega t \cdot \vec{x}_0 + S\omega t \cdot \vec{y}_0) - S\beta \cdot \vec{z}_0 \\ \vec{y}_{11} = -S\omega t \cdot \vec{x}_0 + C\omega t \cdot \vec{y}_0 \\ \vec{z}_{11} = S\beta \cdot (C\omega t \cdot \vec{x}_0 + S\omega t \cdot \vec{y}_0) + C\beta \cdot \vec{z}_0 \end{cases}$$

$$\text{Ou encore : } \begin{cases} \vec{x}_{11} = C\beta \cdot C\omega t \cdot \vec{x}_0 + C\beta \cdot S\omega t \cdot \vec{y}_0 - S\beta \cdot \vec{z}_0 \\ \vec{y}_{11} = -S\omega t \cdot \vec{x}_0 + C\omega t \cdot \vec{y}_0 \\ \vec{z}_{11} = S\beta \cdot C\omega t \cdot \vec{x}_0 + S\beta \cdot S\omega t \cdot \vec{y}_0 + C\beta \cdot \vec{z}_0 \end{cases}$$

$$\text{Au final Pour } \vec{U} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}_{\mathcal{R}_0} \quad \vec{U} = \begin{bmatrix} a'' \\ b'' \\ c'' \end{bmatrix}_{\mathcal{R}_{11}} = \begin{bmatrix} C\beta \cdot C\omega t & C\beta \cdot S\omega t & -S\beta \\ -S\omega t & C\omega t & 0 \\ S\beta \cdot C\omega t & S\beta \cdot S\omega t & C\beta \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}_{\mathcal{R}_c} = R_{y,\beta}^- \times R_{z,\omega t}^- \begin{bmatrix} a \\ b \\ c \end{bmatrix}_{\mathcal{R}_c}$$

Par un calcul manuel, nous pourrions trouver $\vec{U} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}_{\mathcal{R}_0}$ en fonction de $\vec{U} = \begin{bmatrix} a'' \\ b'' \\ c'' \end{bmatrix}_{\mathcal{R}_{11}}$. Utilisons nos chères matrices

$$\text{et leurs propriétés : } \vec{U} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}_{\mathcal{R}_0} = (R_{y,\beta}^- \times R_{z,\omega t}^-)^{-1} \begin{bmatrix} a'' \\ b'' \\ c'' \end{bmatrix}_{\mathcal{R}_{11}} = (R_{z,\omega t}^-)^{-1} \times (R_{y,\beta}^-)^{-1} \begin{bmatrix} a'' \\ b'' \\ c'' \end{bmatrix}_{\mathcal{R}_{11}} = R_{z,-\omega t}^- \times R_{y,-\beta}^- \begin{bmatrix} a'' \\ b'' \\ c'' \end{bmatrix}_{\mathcal{R}_{11}}$$

$$\text{soit } \vec{U} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}_{\mathcal{R}_0} = \begin{bmatrix} C\beta & 0 & S\beta \\ 0 & 1 & 0 \\ -S\beta & 0 & C\beta \end{bmatrix} \times \begin{bmatrix} C\omega t & -S\omega t & 0 \\ S\omega t & C\omega t & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a'' \\ b'' \\ c'' \end{bmatrix}_{\mathcal{R}_{11}}$$

$$\vec{U} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}_{\mathcal{R}_0} = \begin{bmatrix} C\beta \cdot C\omega t & -C\beta \cdot S\omega t & S\beta \\ S\omega t & C\omega t & 0 \\ -S\beta \cdot C\omega t & S\beta \cdot S\omega t & C\beta \end{bmatrix} \begin{bmatrix} a'' \\ b'' \\ c'' \end{bmatrix}_{\mathcal{R}_{11}}$$

Bon « pour le fun » finissons la résolution du problème :

Au niveau du fonctionnement, les deux vecteurs \vec{z}_2 et \vec{z}_{11} sont confondus. Nous pouvons paramétrer la sortie comme suit :

	<p>Avec $\vec{OA} = k \cdot \vec{x}_2 = k(C\lambda \cdot \vec{x}_0 - S\lambda \cdot \vec{z}_0)$. La condition d'orthogonalité de \vec{x}_2 et \vec{z}_2</p> $\begin{cases} \vec{z}_2 = \vec{z}_{11} \\ \vec{x}_2 \perp \vec{z}_2 \Rightarrow \vec{x}_2 \cdot \vec{z}_2 = 0 \end{cases} \text{ donne l'équation scalaire :}$ $(S\beta \cdot C\omega t \cdot \vec{x}_0 + S\beta \cdot S\omega t \cdot \vec{y}_0 + C\beta \cdot \vec{z}_0) \cdot (C\lambda \cdot \vec{x}_0 - S\lambda \cdot \vec{z}_0) = 0, \text{ soit}$ $S\beta \cdot C\omega t \cdot C\lambda - C\beta \cdot S\lambda = 0. \text{ Au final } \text{tg } \lambda = \text{tg } \beta \cdot C\omega t. \text{ Avec } \beta = -\alpha,$ <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> $\text{tg } \lambda = \text{tg } \alpha \cdot \cos(\omega t + \pi)$ </div>
--	--

Le déplacement de la tige de sortie vaut $Z = d \cdot \text{tg}(\lambda) = d \cdot \text{tg}(\alpha) \cdot \cos(\omega t + \pi)$

Hors mis l'exploit cinématique du mécanisme, nous avons touché du doigt la présentation matricielle des changements de coordonnées.

Sur des cas simples comme celui-ci, le recours aux matrices n'est pas forcément une simplification. Dans le cas de mécanismes plus complexes ou dans le cas de la réalisation d'outil de calcul automatique, la systématisation des opérations via les matrices s'avère indispensable.

7.2 Généralisation

Soit deux repères orthonormés $R_a = (\vec{x}_a, \vec{y}_a, \vec{z}_a)$ et $R_b = (\vec{x}_b, \vec{y}_b, \vec{z}_b)$ Un vecteur \vec{V} est représenté par ses

composantes dans les deux repères :
$$\begin{cases} \vec{V} = v_{1,b} \vec{x}_b + v_{2,b} \vec{y}_b + v_{3,b} \vec{z}_b \\ \vec{V} = v_{1,a} \vec{x}_a + v_{2,a} \vec{y}_a + v_{3,a} \vec{z}_a \end{cases}$$
 Synthétiquement nous pouvons dire qu'avec les

repères $R_i = (\vec{x}_i, \vec{y}_i, \vec{z}_i)$, en considérant $v_{k,i} \begin{cases} i : \text{indice du repere } R_i \text{ (ici nous prendrons a ou b)} \\ k : (1, 2, 3) \text{ composante sur l'axe } N^{\circ k} \end{cases}$, nous avons

$$\vec{V} = v_{1,i} \vec{x}_i + v_{2,i} \vec{y}_i + v_{3,i} \vec{z}_i .$$

Matriciellement nous écrivons :
$$\begin{bmatrix} v_{1,a} \\ v_{2,a} \\ v_{3,a} \end{bmatrix}_{R_a} = \mathbf{P}_{a,b} \times \begin{bmatrix} v_{1,b} \\ v_{2,b} \\ v_{3,b} \end{bmatrix}_{R_b} ; \mathbf{P}_{a,b} = [p_{i,j}]_{3 \times 3} ; p_{i,j} = \vec{a}_{i,a} \cdot \vec{a}_{j,b} \text{ avec } \vec{a}_{i,f} \text{ le vecteur de}$$

base de rang i de la base f . Pour mémoire on notera aussi : $\vec{a}_1 = \vec{x}, \vec{a}_2 = \vec{y}, \vec{a}_3 = \vec{z}$.

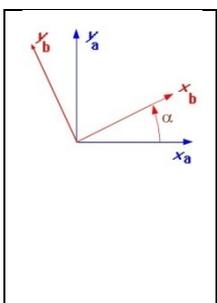
La matrice de transformation de composantes $\mathbf{P}_{a,b}$ est décrite comme suit :

$$\mathbf{P}_{a,b} = \begin{bmatrix} \vec{x}_a \cdot \vec{x}_b & \vec{x}_a \cdot \vec{y}_b & \vec{x}_a \cdot \vec{z}_b \\ \vec{y}_a \cdot \vec{x}_b & \vec{y}_a \cdot \vec{y}_b & \vec{y}_a \cdot \vec{z}_b \\ \vec{z}_a \cdot \vec{x}_b & \vec{z}_a \cdot \vec{y}_b & \vec{z}_a \cdot \vec{z}_b \end{bmatrix}$$

On remarque que dans la matrice $\mathbf{P}_{a,b}$, nous trouvons par lignes les composantes des vecteurs de base de la base a

$$\text{exprimé dans la base } b : \mathbf{P}_{a,b} = \begin{bmatrix} \vec{x}_a \cdot \vec{x}_b & \vec{x}_a \cdot \vec{y}_b & \vec{x}_a \cdot \vec{z}_b \\ \vec{y}_a \cdot \vec{x}_b & \vec{y}_a \cdot \vec{y}_b & \vec{y}_a \cdot \vec{z}_b \\ \vec{z}_a \cdot \vec{x}_b & \vec{z}_a \cdot \vec{y}_b & \vec{z}_a \cdot \vec{z}_b \end{bmatrix} \equiv \begin{bmatrix} \text{composantes de } \vec{x}_a \text{ dans } R_b \\ \text{composantes de } \vec{y}_a \text{ dans } R_b \\ \text{composantes de } \vec{z}_a \text{ dans } R_b \end{bmatrix}$$

Pour se persuader de la réalité de ces formules, vérifions les dans un cas particulier ou le repère b se déduit du repère a par une rotation de α autour de l'axe $\vec{z}_a = \vec{z}_b$



$$\begin{cases} \vec{x}_b = C\alpha \cdot \vec{x}_a + S\alpha \cdot \vec{y}_a \\ \vec{y}_b = -S\alpha \cdot \vec{x}_a + C\alpha \cdot \vec{y}_a \\ \vec{z}_b = \vec{z}_a \end{cases} \text{ Avec } \vec{V} = p\vec{x}_b + q\vec{y}_b + r\vec{z}_b \text{ nous trouvons :}$$

$$\vec{V} = p(C\alpha \cdot \vec{x}_a + S\alpha \cdot \vec{y}_a) + q(-S\alpha \cdot \vec{x}_a + C\alpha \cdot \vec{y}_a) + r \cdot \vec{z}_a ;$$

$$\vec{V} = (p \cdot C\alpha - q \cdot S\alpha) \vec{x}_a + (p \cdot S\alpha + q \cdot C\alpha) \vec{y}_a + r \cdot \vec{z}_a .$$

Matriciellement il vient :

$$\begin{bmatrix} a \cdot C\alpha - b \cdot S\alpha \\ a \cdot S\alpha + b \cdot C\alpha \\ r \end{bmatrix}_{R_a} = \mathbf{P}_{a,b} \begin{bmatrix} p \\ q \\ r \end{bmatrix}_{R_b} = \begin{bmatrix} C\alpha & -S\alpha & 0 \\ S\alpha & C\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}_{R_b} \text{ on constate que :}$$

