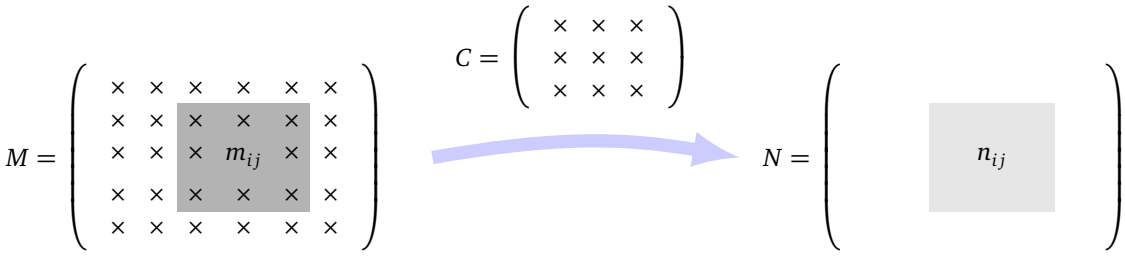
Images et matrices

Le traitement des images est très utile, par exemple pour les agrandir ou bien les tourner. Nous allons aussi voir comment rendre une image plus floue, mais aussi plus nette ! Tout cela à l'aide des matrices.

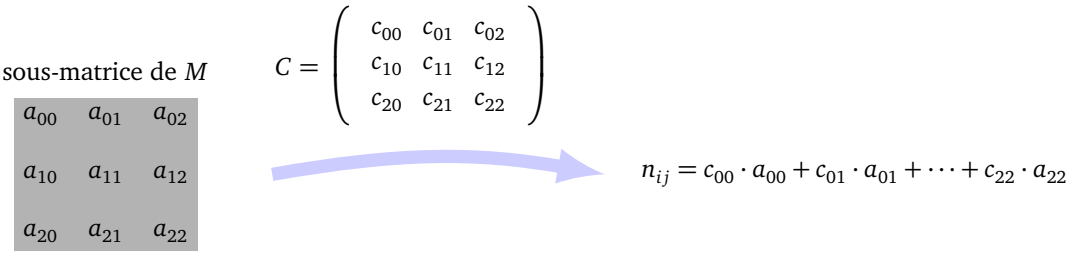
Cours 1 (Convolution de matrices)**.**
La convolution est l'action d'une matrice C de taille 3×3 sur un matrice M de taille quelconque et qui transforme la matrice M en une matrice N de même taille.

- **Principe.** Pour chaque élément m_{ij} de la matrice M de départ, on regarde la sous-matrice 3×3 qui entoure cet élément (en gris foncé ci-dessous), on va multiplier ces éléments par les éléments de la matrice C, tout ceci va donner un seul élément : le coefficient n_{ij} de la matrice N.

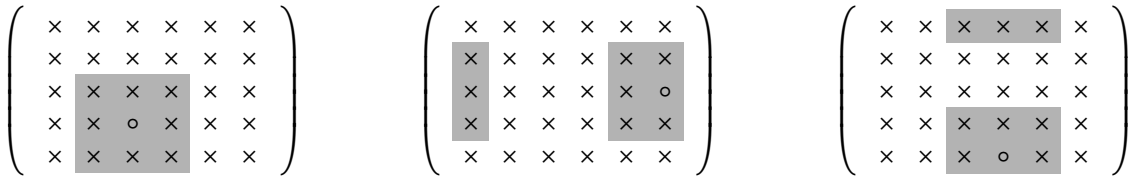


- **Élément de convolution.** Voici comment calculer un élément de convolution. On multiplie chaque élément de la sous-matrice 3×3 de M par l'élément correspondant de C. La somme de ses produits donne l'élément de convolution :

$$n_{ij} = c_{00}a_{00} + c_{01}a_{01} + \cdots + c_{22}a_{22}$$



- **Matrice obtenue.** On calcule donc la matrice N coefficient par coefficient, à chaque coefficient (symbolisés par un cercle) correspond donc une sous matrice 3×3 de la matrice M (figure ci-dessous à gauche). Comment faire pour les coefficients qui sont au bord de la matrice ? Si la sous-matrice déborde à droite elle repart à gauche (figure du centre), si elle déborde en bas, elle repart en haut, etc.



Si on ne tenait pas compte des contraintes de bords, les coefficients de la sous-matrice autour du coefficient m_{ij} seraient donnés par la sous-matrice ci-dessous à gauche, mais pour tenir compte des débordements on raisonne à l'aide des modulos (sous-matrice de droite). Les indices i des lignes sont calculés modulo n (où n est le nombre de lignes de la matrice M), les indices j des colonnes sont calculés modulo p (où p est le nombre de colonnes de la matrice M) :

$m_{i-1,j-1}$	$m_{i-1,j}$	$m_{i-1,j+1}$
$m_{i,j-1}$	$m_{i,j}$	$m_{i,j+1}$
$m_{i+1,j-1}$	$m_{i+1,j}$	$m_{i+1,j+1}$

$m_{(i-1)\%n,(j-1)\%p}$	$m_{(i-1)\%n,j}$	$m_{(i-1)\%n,(j+1)\%p}$
$m_{i,(j-1)\%p}$	$m_{i,j}$	$m_{i,(j+1)\%p}$
$m_{(i+1)\%n,(j-1)\%p}$	$m_{(i+1)\%n,j}$	$m_{(i+1)\%n,(j+1)\%p}$

- Exemple.

$$M = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix} \qquad C = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad N = \begin{pmatrix} 22 & 24 & 30 & 32 \\ 34 & 36 & 42 & 44 \\ 46 & 48 & 54 & 56 \end{pmatrix}$$

Par exemple voici le calcul du coefficient n_{11} :

$$n_{11} = c_{00}m_{00}+c_{01}m_{01}+\cdots+c_{22}m_{22} = 0\times1+1\times2+0\times3+1\times5+2\times6+1\times7+0\times9+1\times10+0\times11 = 36$$

Cours 2 (Modélisation de matrices)**.**

- Une matrice est codée par un tableau, c'est-à-dire une liste de listes. Par exemple :

```
M = [[1,2,3,4], [5,6,7,8], [9,10,11,12]]
```

représente la matrice :

$$M = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

- On accède à l'élément (i, j) par :

```
M[i][j]
```

- Une matrice M de n lignes et p colonnes est donc une liste de longueur n, chaque élément étant une liste de longueur p :

```
n = len(M)     p = len(M[0])
```

- Comme d'habitude la numérotation commence à 0, donc $0 \leqslant i < n$ et $0 \leqslant j < p$.
- On initialise une matrice avec des coefficients nuls ainsi :

```
M = [[0 for j in range(p)] for i in range(n)]
```

On peut ensuite remplir la matrice coefficient par coefficient par des commandes du type :

```
M[i][j] = ...
```

Activité 1 (Convolution de matrices)**.**

Objectifs : calculer une convolution.

1. Programme une fonction `afficher_matrice(M)` qui réalise un affichage propre d'une matrice sur le terminal de l'écran.

Voici un exemple d'affichage :

```
M = [[1,2,3], [4,5,6], [7,8,9]]
```

```
  1   2   3
  4   5   6
  7   8   9
```

Indications.

- `print('{:3d}'.format(x),end=" ")` affiche un entier sur 3 caractères.
- `print('{0:.3f}'.format(x),end=" ")` affiche un nombre flottant avec 3 décimales après la virgule.

2. Programme une fonction `element_convolution(C,M)` qui à partir de deux matrices C et M de taille 3×3 calcule l'élément de convolution :

$$\gamma = c_{00}m_{00} + c_{01}m_{01} + \cdots + c_{22}m_{22}$$

Autrement dit γ est la somme des produits des coefficients entre C et M. Par exemple pour :

```
C = [[1,1,1], [1,5,1], [1,1,1]]
M = [[1,2,3], [4,5,6], [7,8,9]]
```

L'élément de convolution est

$$\gamma = 1\times1+1\times2+1\times3+1\times4+5\times5+1\times6+1\times7+1\times8+1\times9 = 65.$$

3. Programme une fonction `convolution(C,M)` qui calcule la matrice N de convolution de C sur M.
   - C est une matrice 3×3.
   - M est une matrice de taille quelconque $n \times p$.
   - $N = \text{convolution}(C, M)$ est aussi de taille $n \times p$ et chaque coefficient n_{ij} est l'élément de convolution de C avec M'_{ij} la sous-matrice 3×3 de M autour du coefficient en (i, j).
   - N'oublie pas que pour la sous-matrice il y a des effets de bords (voir les formules avec les modulos dans le cours ci-dessus).

   Par exemple

$$C = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad M = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix} \qquad \text{donnent} \qquad N = \begin{pmatrix} 9 & 10 & 11 & 12 \\ 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

   Avec cette matrice C, la convolution sur M correspond à faire descendre chaque coefficient d'une ligne (et comme on travaille avec les modulos, un coefficient de la dernière ligne se retrouve sur la première ligne).

4. Utilise ou modifie la fonction précédente en une fonction `convolution_entiere(C,M)` qui renvoie une matrice dont les coefficients sont arrondis à des entiers, limités à 255, les coefficients négatifs étant ramenés à zéros.

   Exemple.

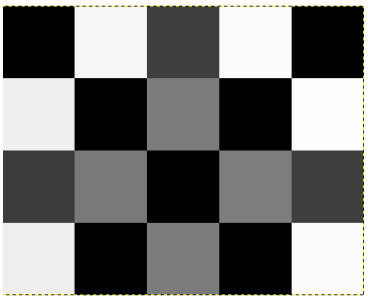
$$C = \begin{pmatrix} 0 & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 3 & -\frac{1}{2} \\ 0 & -\frac{1}{2} & 0 \end{pmatrix} \qquad M = \begin{pmatrix} 101 & 102 & 103 & 104 \\ 201 & 151 & 101 & 51 \\ 50 & 100 & 150 & 200 \end{pmatrix}$$

donnent la convolution N et la convolution entière N' :

$$N = \begin{pmatrix} 74.5 & 78.5 & 80.5 & 84.5 \\ 426.5 & 201.0 & 75.5 & -150.0 \\ -151.0 & 73.5 & 198.0 & 422.5 \end{pmatrix} \quad \text{et} \quad N' = \begin{pmatrix} 74 & 78 & 80 & 84 \\ 255 & 201 & 75 & 0 \\ 0 & 73 & 198 & 255 \end{pmatrix}$$

Cours 3 (Format d'image « pgm »)**.**
Le format d'image « pgm » est un format très simple de fichier texte qui permet de décrire une image. Voici un exemple très simple : à gauche le fichier et à droite l'image correspondante.

```
P2
5 4
255
  0 255  64 255   0
255   0 127   0 255
 64 127   0 127  64
255   0 127   0 255
```



Explications.

- `P2` est le code pour le format d'image en niveau de gris (`P1` désigne les images en noir et blanc, `P3` les images en couleurs).
- Les entiers 5 et 4 sont le nombre de colonnes suivi du nombre de lignes de l'image (ici 5 colonnes et 4 lignes).
- 255 désigne le découpage en niveau de gris : ici de 0 (noir) à 255 (blanc).
- Les entiers suivants sont le niveau de gris de chacun des pixels de l'image.

Version étendue. Ce format d'image est reconnu par tous les bons lecteurs d'images. Les logiciels de retouches d'images permettent d'exporter n'importe quelle image vers ce format. Par contre la structure du fichier peut être différente : tout d'abord il peut y avoir des lignes de commentaires (commençant par #), ensuite les valeurs de niveau de gris de chaque pixel peuvent être toutes sur une même ligne ou bien un seul pixel par ligne. C'est par exemple le cas du logiciel *gimp* qui produit des fichiers comme ci-dessous à gauche.

```
P2
# Commentaire !
5 4
255
0
255
64
255
0
255
...
```

```
P2
5 4
255
0 255 64 255 0 255 0 127 ...
```

Activité 2 (Lire et écrire des images)**.**

Objectifs : convertir un fichier d'image en une matrice et inversement.

1. Programme une fonction `pgm_vers_matrice(fichier)` qui lit un fichier d'image au format « pgm » dont le nom est donné et renvoie une matrice.
   Voici un exemple de fichier et la matrice associée :

```
P2
4 5
255
  0   0   0   0
  0 255 255   0
  0 128 128   0
255 255 255 255
  0  64  64   0
```

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 255 & 255 & 0 \\ 0 & 128 & 128 & 0 \\ 255 & 255 & 255 & 255 \\ 0 & 64 & 64 & 0 \end{pmatrix}$$

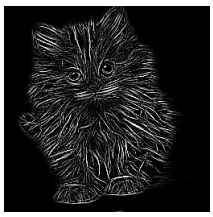
   C'est préférable si ta fonction accepte les fichiers au format « pgm » étendu qui peuvent contenir des commentaires et où il n'y a qu'une nuance de gris par ligne. (Voir le cours ci-dessus.)
2. Fais le travail inverse en programmant une fonction `matrice_vers_pgm(M,fichier)` qui part d'une matrice M et écrit un fichier (dont le nom est donné) au format « pgm ».

Activité 3 (Convolution d'une image)**.**

Objectifs : appliquer la convolution pour transformer une image.


Image originale


Flou


Bords


Piqué


Estampé

Programme une fonction `convolution_image(C,fichier_in,fichier_out)` qui à partir d'une matrice de convolution C de taille 3×3 et d'une image (dont le nom est donné) écrit les éléments d'une nouvelle image dans un fichier (dont le nom est aussi donné). Les fichiers d'images sont au format « pgm ».
C'est une fonction très simple qui utilise la fonction `convolution_entiere()`. Ensuite teste différentes convolutions !

Flou. La convolution avec la matrice suivante « floute » légèrement l'image. C'est parce que le niveau de gris de chaque nouveau pixel est la moyenne des niveaux des 9 pixels voisins de l'image de départ.

$$C = \begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$$

Image originale

Flou

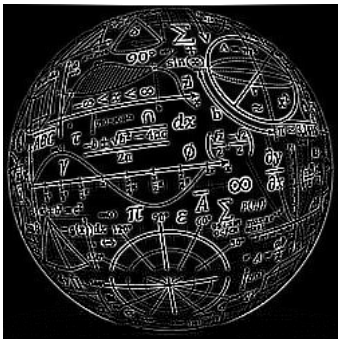
Essaye aussi le *flou gaussien* avec la matrice :

$$C = \begin{pmatrix} \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \\ \frac{2}{16} & \frac{4}{16} & \frac{2}{16} \\ \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \end{pmatrix}$$

Bords.

$$C = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

Image originale



Bords

Essaie aussi les matrices :

$$C = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad \text{ou} \quad C = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

Piqué.
C'est le contraire du flou !

$$C = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Voici un exemple avec l'image originale à gauche et l'image transformée à droite qui a l'air plus nette que l'originale !

Image originale

Piqué

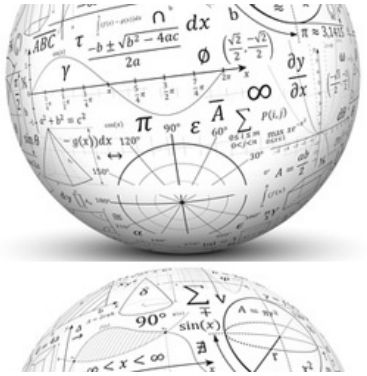
Estampé.

$$C = \begin{pmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

Image originale

Estampé

Décalage vers le haut.
Trouve la convolution qui décale l'image d'un pixel vers le haut et permet d'obtenir l'image suivante par itération.



Décalage itéré vers le haut

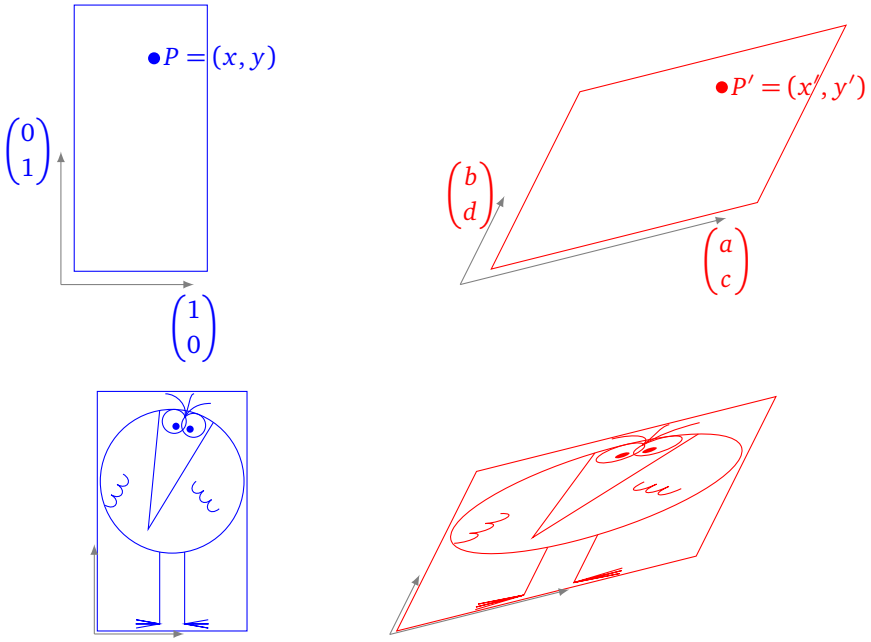
Cours 4 (Matrice de transformation)**.**
Voici de brefs rappels sur les matrices en se limitant aux matrices 2×2. Pour l'interprétation géométrique, on identifie un point P de coordonnées (x, y) avec le vecteur $\vec{u} = \begin{pmatrix} x \\ y \end{pmatrix}$.

- **Multiplication par un vecteur.**

$$X = \begin{pmatrix} x \\ y \end{pmatrix} \quad \text{et} \quad M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \qquad \text{alors} \qquad X' = AX = \begin{pmatrix} ax+by \\ cx+dy \end{pmatrix}$$

Autrement dit, les coordonnées (x', y') de l'image de $P = (x, y)$ sont données par :

$$\begin{cases} x' &= ax+by \\ y' &= cx+dy \end{cases}$$

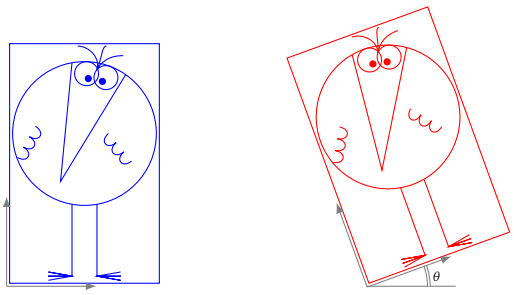


- **Matrice de rotation.**

$$T_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

Autrement dit, pour $X' = T_\theta X$, on a :

$$\begin{cases} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{cases}$$



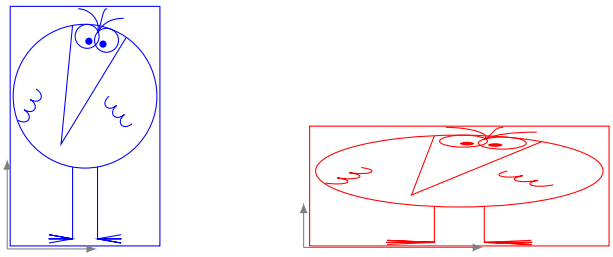
- **Dilatation.**

$$T = \begin{pmatrix} k_x & 0 \\ 0 & k_y \end{pmatrix}$$

Autrement dit :

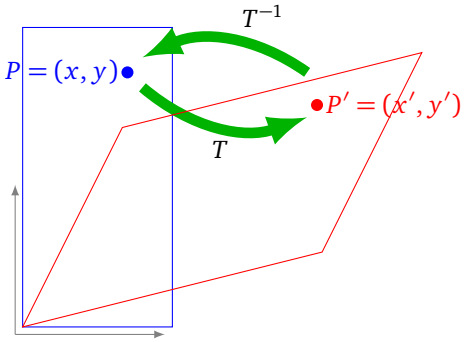
$$\begin{cases} x' &= k_x \cdot x \\ y' &= k_y \cdot y \end{cases}$$

Un exemple avec $k_x = 2$, $k_y = \frac{1}{2}$.



- **Inverse.** Une matrice M, dont le déterminant est non nul, admet un inverse M^{-1} :

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \qquad M^{-1} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$



Activité 4 (Transformation d'images)**.**

Objectifs : utiliser les matrices pour tourner, inverser ou déformer une image.

1. **Dilatation.**

   Programme une fonction `dilatation_matrice(kx,ky,M)` qui effectue une dilatation de la matrice M. La matrice est agrandie d'un facteur k_x horizontalement et d'un facteur k_y verticalement en une nouvelle matrice N (ici k_x et k_y sont des entiers).

   Exemple. Voici un exemple avec $k_x = 3$ et $k_y = 2$.

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad \text{devient} \qquad N = \begin{pmatrix} 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 \\ 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 \\ 4 & 4 & 4 & 5 & 5 & 5 & 6 & 6 & 6 \\ 4 & 4 & 4 & 5 & 5 & 5 & 6 & 6 & 6 \\ 7 & 7 & 7 & 8 & 8 & 8 & 9 & 9 & 9 \\ 7 & 7 & 7 & 8 & 8 & 8 & 9 & 9 & 9 \end{pmatrix}$$

   Formule. On obtient les coefficients de la nouvelle matrice N en fonction des coefficients de l'ancienne matrice M :

$$N[i][j] = M[i//k_y][j//k_x]$$

   Indications.

   - Commence par initialiser une matrice vide :

     ```
     N = [[0 for j in range(kx*p)] for i in range(ky*n)]
     ```

     de la bonne taille.
   - Prends garde que horizontalement x correspond à j et verticalement y correspond à i.

   Application. Dilate des images.

Image originale

Image dilatée ($k_x = 2, k_y = 3$)

L'image dilatée est plus grande mais sa résolution est plus faible, au final il n'y a ni gain ni perte d'information.

2. **Matrice de transformation.**

   On souhaite généraliser la question précédente avec une transformation quelconque.

   - Programme une fonction `vecteur_image(T,x,y)` qui à partir d'une matrice de transformation T et des coordonnées $\binom{x}{y}$ d'un vecteur renvoie les coordonnées $\binom{x'}{y'}$ de l'image de ce vecteur par T :
   $$T = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \qquad \begin{pmatrix} x' \\ y' \end{pmatrix} = T \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ax + by \\ cx + dy \end{pmatrix}$$
   - Programme une fonction `inverse_matrice(T)` qui renvoie la matrice inverse de la matrice T (dont on suppose le déterminant non nul).
   $$T = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \qquad T^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$
   - *Application.* Soit T_θ la matrice de rotation d'angle θ :
   $$T_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$
   Soit $\theta = \frac{\pi}{3}$. Soit $P = (x, y) = (4, 5)$. Calcule l'image de P par la rotation d'angle θ. C'est le point $P' = (x', y')$ dont les coordonnées sont obtenues par multiplication de T_θ par le vecteur $\binom{x}{y}$.
   - *Application.* Vérifie pour $\theta = \frac{\pi}{3}$ (par exemple) que la matrice de rotation $T_{-\theta}$ est égale à la matrice $(T_\theta)^{-1}$. (C'est juste une vérification par le calcul que l'opération inverse de tourner d'un angle θ c'est tourner d'un angle $-\theta$!)

3. **Action de la transformation.**

   Le but est de programmer une fonction `transformation(T,M)` qui à partir d'une matrice T de taille 2×2 transforme une matrice M (de taille quelconque) en une nouvelle matrice N.

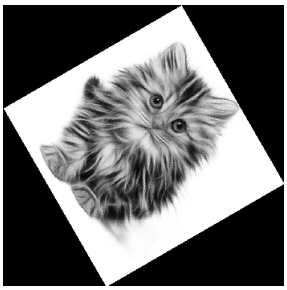
   Application. Tu peux ensuite faire agir n'importe quelle matrice T pour transformer une image.

Dilatation (avec constantes quelconques).

$$T = \begin{pmatrix} k_x & 0 \\ 0 & k_y \end{pmatrix} \quad \text{avec } k_x = \pi = 3.14\ldots,\ k_y = 1$$

Rotation.

$$T = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \qquad \text{avec } \theta = \frac{\pi}{3}$$



Symétrie.

$$T = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Transformation quelconque.

$$T = \begin{pmatrix} 3 & 1 \\ 1 & 5 \end{pmatrix}$$

Voici comment programmer cette fonction.

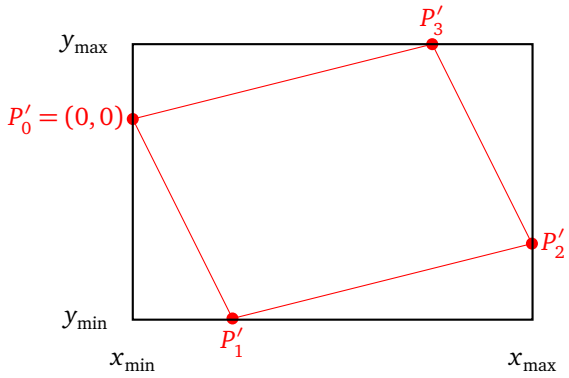
Première étape : taille de la nouvelle matrice.

On identifie la matrice M de taille $n \times p$ à une image rectangulaire. Les quatre coins de cette image ont pour coordonnées $P_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $P_1 = \begin{pmatrix} p \\ 0 \end{pmatrix}$, $P_2 = \begin{pmatrix} p \\ n \end{pmatrix}$, $P_3 = \begin{pmatrix} 0 \\ n \end{pmatrix}$. Calcule les coordonnées $P_i' = \begin{pmatrix} x_i \\ y_i \end{pmatrix} = T \cdot P_i$ des coins transformés : ce sont les sommets d'un parallélogramme (ce n'est pas toujours un rectangle).

On note :

$$x_{\min} = \min(x_0, x_1, x_2, x_3) \qquad \text{et} \qquad x_{\max} = \max(x_0, x_1, x_2, x_3)$$
$$y_{\min} = \min(y_0, y_1, y_2, y_3) \qquad \text{et} \qquad y_{\max} = \max(y_0, y_1, y_2, y_3)$$

L'image transformée est incluse dans le rectangle défini par $(x_{\min}, x_{\max}, y_{\min}, y_{\max})$!



On retient :

- le nombre de lignes de la matrice transformée est $n' = y_{\max} - y_{\min}$ (c'est la hauteur de l'image transformée),
- le nombre de colonnes de la matrice transformée est $p' = x_{\max} - x_{\min}$ (c'est la largeur de l'image transformée).

Seconde étape : formule de la transformation.

La nouvelle matrice N sera une matrice de taille $n' \times p'$.

Pour savoir combien vaut le coefficient $N[i'][j']$, il faut trouver le coefficient $M[i][j]$ correspondant dans la matrice de départ. (Autrement dit : la nouvelle l'image est de taille $n' \times p'$, le pixel (i', j') est de niveau de gris $N[i'][j']$ qui se calcule en fonction du niveau de gris $M[i][j]$.)

Il faut donc utiliser la matrice inverse T^{-1}. En plus il faut effectuer une translation de vecteur $\binom{x_{\min}}{y_{\min}}$ pour recentrer l'origine. Ce qui donne :

$$\begin{pmatrix} j \\ i \end{pmatrix} = T^{-1} \begin{pmatrix} j' + x_{\min} \\ i' + y_{\min} \end{pmatrix}$$

et ensuite simplement :

$$N[i'][j'] = M[i][j]$$

Exemple. Voici une dilatation avec des coefficients non entiers : le nombre de colonnes passe de 3 à 4 (facteur $k_x = \frac{4}{3}$), le nombre de lignes est doublé ($k_y = 2$).

$$T = \begin{pmatrix} \frac{4}{3} & 0 \\ 0 & 2 \end{pmatrix} \qquad M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad N = \begin{pmatrix} 1 & 1 & 2 & 3 \\ 1 & 1 & 2 & 3 \\ 4 & 4 & 5 & 6 \\ 4 & 4 & 5 & 6 \\ 7 & 7 & 8 & 9 \\ 7 & 7 & 8 & 9 \end{pmatrix}$$

Tu peux maintenant transformer des images.