

TPs EDP ^a

Travaux Pratiques N° 1

• • • • •

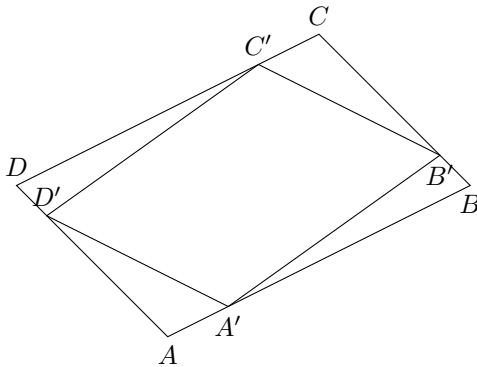
Algorithmique : mosaïques

^a. Version du 23 septembre 2024

Remarques

- Le premier exercice est **à faire** sur feuille (durée 30mn) et à rendre.
- Dans tous les exercices il faudra user (voir abuser) de la création de fonctions.
- On abusera aussi de **namespace** pour en maîtriser l'usage.

EXERCICE 1



Soit un parallélogramme de sommets A , B , C et D .
A partir de ce parallélogramme on peut construire un nouveau parallélogramme de sommets A' , B' , C' et D' vérifiant

$$\overrightarrow{AA'} = x\overrightarrow{AB}$$

$$\overrightarrow{BB'} = x\overrightarrow{BC}$$

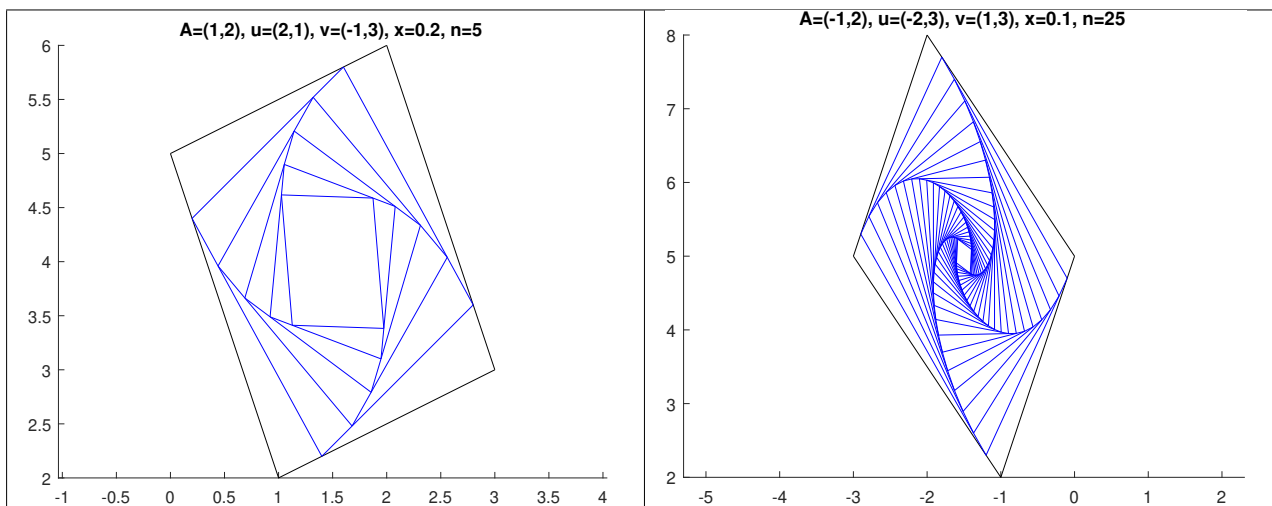
$$\overrightarrow{CC'} = x\overrightarrow{CD}$$

$$\overrightarrow{DD'} = x\overrightarrow{DA}$$

avec x un réel compris strictement entre 0 et 1.

L'objectif est, pour un x fixé, d'itérer n fois ce processus de construction en partant à chaque itération du dernier parallélogramme construit et de représenter l'ensemble des parallélogrammes.

Q. 1 Ecrire une fonction algorithmique *parallélogrammes* permettant à partir du sommet A , du vecteur $\mathbf{u} = \overrightarrow{AB}$ et du vecteur $\mathbf{v} = \overrightarrow{AD}$ d'un parallélogramme initial quelconque d'aire non nulle, de représenter ce parallélogramme ainsi que les n parallélogrammes obtenus par le processus de construction décrit ci-dessus avec un x donné dans $]0, 1[$. On dispose pour cela de la fonction `plot([xA, xB], [yA, yB])` permettant de tracer le segment $[A, B]$ du plan avec $A = (x_A, y_A)$ et $B = (x_B, y_B)$. Voici deux exemples d'utilisation de cette fonction :



□

Instructions pour la suite

Deux archives compressées de même contenu au format **zip**

<https://www.math.univ-paris13.fr/~cuvelier/docs/Enseignements/MACS2/TPs-EDP/24-25/TP1.zip>

et au format **tar.gz**

<https://www.math.univ-paris13.fr/~cuvelier/docs/Enseignements/MACS2/TPs-EDP/24-25/TP1.tar.gz>

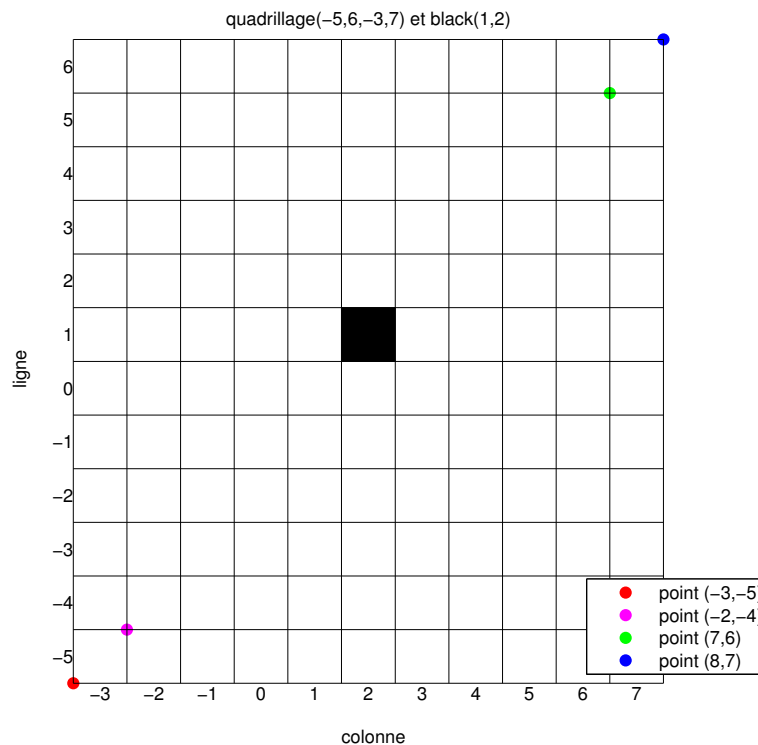
sont disponibles en ligne. Il faut télécharger une des archives et la décompresser dans un répertoire. Cette archive contient la fonction `exo2.PrintNumbers` et le programme `exo2.Quadrillagefigure`. Dans le programme `exo2.Quadrillagefigure` les appels aux fonctions manquantes `exo2.black` et `exo2.Quadrillage` ont été mis en commentaire. Ce programme va vous permettre, en autres, de valider/tester la fonction `exo2.Quadrillage` que vous allez écrire.

EXERCICE 2

Ecrire les fonctions graphiques `exo2.Quadrillage` et `exo2.black` :

- la fonction `exo2.Quadrillage(imin,imax,jmin,jmax)` permet de générer un quadrillage pour les lignes `imin` à `imax` et les colonnes `jmin` à `jmax`. Cette fonction trace uniquement les traits noirs horizontaux et verticaux du quadrillage (utilisation de la fonction `plot` de Matlab).
- la fonction `exo2.black(i,j)` remplit en noir la *case* en ligne `i` et colonne `j` du quadrillage (utilisation de la fonction `fill` de Matlab).

Voici un exemple d'utilisation de la commande `exo2.Quadrillage(-5,6,-3,7)` générant uniquement les traits noirs de la figure :



On peut noter que les coordonnées des points sont exprimées dans le plan classique xOy . On doit tester/valider cette fonction avec le programme `exo2.Quadrillagefigure` fourni pour obtenir la figure précédente. Le carré noir en ligne 1 et colonne 2 a été représenté à l'aide de la commande `exo2.black(1,2)`. La numérotation des lignes et des colonnes a été réalisée par la fonction `exo2.PrintNumbers` fournie dans l'archive.

Il est primordial que ces deux fonctions soient parfaitement valides pour la suite du TP !

EXERCICE 3

Q. 1 On souhaite écrire la fonction `exo3.PrintNumbers` qui sera une réécriture de la fonction `exo2.PrintNumbers` avec des paramètres optionnels en utilisant la classe `inputParser` :

- `'steprow'` permettra de définir le nombre de pas entre deux affichages consécutifs des numéros de lignes (valeur par défaut 2),
- `'stepcol'` idem pour les colonnes,
- ainsi que tous les paramètres optionnels de la fonction `text` utilisée (modification fonte, couleur, ...)

Par exemple, voici des exemples d'appels possible :

```
exo3.PrintNumbers(1,20,1,20)
exo3.PrintNumbers(1,20,1,20,'stepcol',3)
exo3.PrintNumbers(1,20,1,20,'steprow',3,'stepcol',3,'fontsize',12)}
```

□

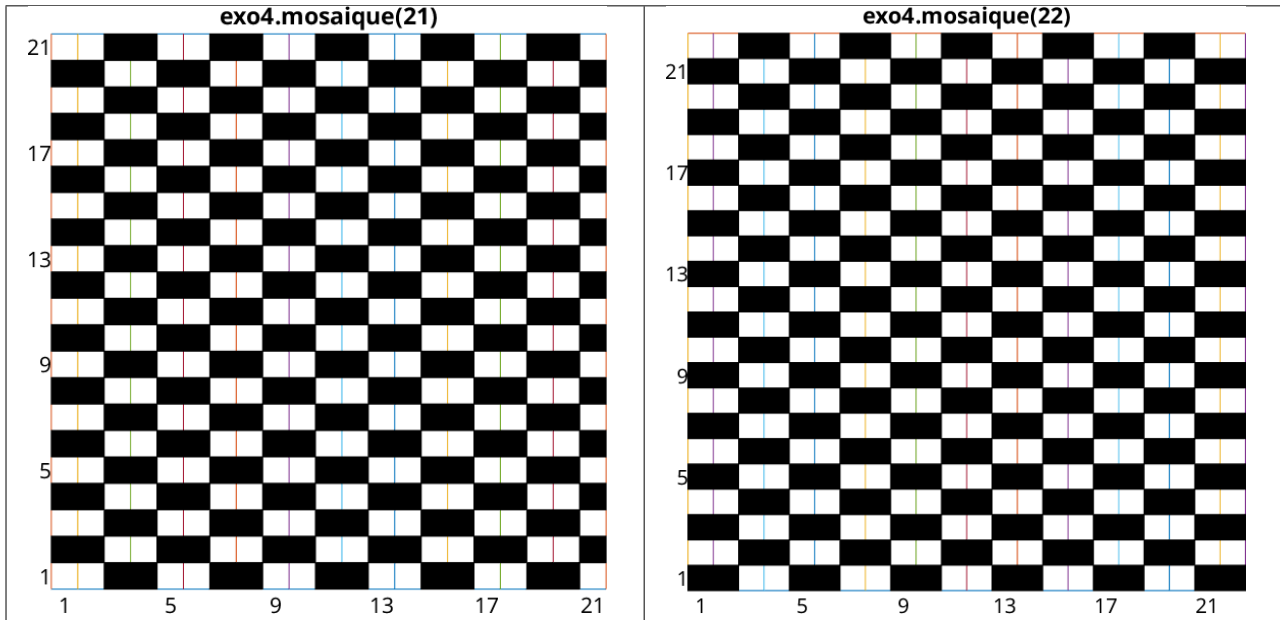
Q. 2 On souhaite écrire la fonction `exo3.Quadrillage` qui sera une réécriture de la fonction `exo2.Quadrillage` avec comme paramètres optionnels ceux de la fonction `plot` utilisée. Par exemple, voici des exemples d'appels possible :

```
exo3.Quadrillage(1,20,1,30)
exo3.Quadrillage(1,20,1,30,'color','r')
exo3.Quadrillage(1,20,1,30,'color',[1,0,0],'linewidth',2)}
```

□

EXERCICE 4

En utilisant les fonctions `exo3.Quadrillage`, `exo3.PrintNumbers` et `exo2.black`, écrire une fonction `exo4.mosaïque` de paramètre $n \in \mathbb{N}$, $n \geq 5$, permettant de créer des figures sur le quadrillage de lignes 1 à n et de colonnes 1 à n . Voici deux exemples, avec $n = 21$ et $n = 22$, des figures que l'on souhaite représenter sachant que la ligne n suit nécessairement la séquence *blanc, blanc, noir, noir* - *blanc, blanc, noir, noir* - , ...



EXERCICE 5

En utilisant les fonctions `exo3.Quadrillage`, `exo3.PrintNumbers` et `exo2.black`, écrire une fonction `exo5.mosaïque` de paramètre $n \in \mathbb{N}$, $n \geq 3$, permettant de créer des figures sur le quadrillage de lignes $-n$ à n et de colonnes $-n$ à n . Voici deux exemples, avec $n = 13$ et $n = 14$, des figures que l'on souhaite représenter :

