


# Py2tikz

Py2tikz  est un outil d'export de figure matplotlib en document tikz/pgfplots développé au laboratoire.

## Objectifs

L'objectif de l'outil est de permettre un export rapide de figures matplotlib de façon automatique. Les figures cibles sont les figures relativement simples et classiquement utilisées :

- graphes 2D
- graphes 3D
- graphes en couleur (contourplot)
- histogrammes
- tracés dans le repère polaire
- combinaison des plusieurs types de graphes

En effet, afin de garder l'outil assez simple et avec des paramètres d'entrée relativement limités, le code ne pourra pas traiter les graphes les plus compliqués. Pour ces derniers, il est toujours possible d'utiliser Py2tikz pour faire une ébauche de la figure finale sans les éléments limitants, puis rajouter manuellement ces derniers.

## Paramètres d'entrée

Les options d'entrée (commit 64019c27a35da08117405e26e4b850f8f155c067) sont :

### Obligatoires:

- `fig` (type : `matplotlib.pyplot.Figure`) : une figure à exporter , par défaut `fig = plt.gcf()` (figure courante)
- `tex_file_name` (type : `str`) : nom du fichier `.tex` à enregistrer, par défaut `tex_file_name = 'py2tikz_fig'`
- `rel_path_to_main_file` (type : `str`) : chemin relatif depuis le `current working directory` (`os.getcwd()`) jusqu'au dossier d'enregistrement, par défaut `rel_path_to_main_file = ''` : chaîne de caractère vide, donc au même endroit que l'exécution du code

### Optionnels:

- `width` (type : `str`) : largeur de l'axe , par défaut `width = '6cm'`
- `height` (type : `str`) : hauteur de l'axe , par défaut `height = '6cm'`
- `legend` (type : `bool`) : activer la légende , par défaut `legend = True`. Pour l'instant la légende est affichée en dehors de la figure dans le coin haut droit (à modifier)
- `grid` (type : `bool`) : activer la grille , par défaut `grid = False`. Pour l'instant la seule option est une grille sur les ticks majeurs.

- `auto_xticks` (type : bool) : utiliser les xticks automatiques de tikz , par défaut `auto_xticks = False`.
- `auto_yticks` (type : bool) : utiliser les yticks automatiques de tikz , par défaut `auto_yticks = False`.
- `type_axis`(type : str) : type d'axe à utiliser . Par défaut `type_axis = 'axis'` , autres options : `semilogxaxis`, `semilogyaxis`, `loglogaxis`.
- `RAZ_figure` (type : bool) : Remise à zéro du dossier de la figure pour ne pas stocker tous les fichiers data des exécutions précédentes. Par défaut `RAZ_figure = False`.
- `split_subplots` (type : bool) : Exportation de chaque subplot dans un fichier pdf séparé. Par défaut `split_subplots = False`.
- `opacity` (type : bool) : Prise en charge de l'opacité dans le graphe (surfaces remplies), pour des soucis d'économie mémoire. Par défaut `opacity = False`.
- `mappable` (type : bool) : Prise en charge d'un élément de remplissage (contourplot dans le graphe) qui sera enregistré en image et mis en fond de l'axe correspondant, par défaut `mappable = False`.
- `compileur` (type : str) : Compileur latex utilisé pour la compilation du .tex, à utiliser dans le cas d'une figure avec beaucoup de données en utilisant `opacity = 'lualatex'` , par défaut `opacity = 'pdflatex'` .
- `type_figure` (type : str) : type de figure traitée, spécifier `type_figure = '3D'` dans le cas d'une figure avec trois axes, par défaut `type_figure = '2D'`

## Utilisation

L'utilisation du code est assez simple :

1. Installer la librairie, dans le dossier `py2tikz` (qui contient `setup.py`) dans une console :

```
pip install .
```

1. Importer la fonction `py2tikz` du module `py2tikz` (depuis n'importe où grâce à l'installation par pip du module) :

```
from py2tikz import py2tikz
```

1. Appeler la fonction `py2tikz` par exemple :

```
py2tikz(fig, 'Nom_du_fichier_tex', height='6cm', width='10cm', grid=True)
```

On trouvera alors un nouveau dossier 'tikz' qui contient la figure sous l'arborescence :

`tikz/Nom_du_fichier_tex/Nom_du_fichier_tex.tex`

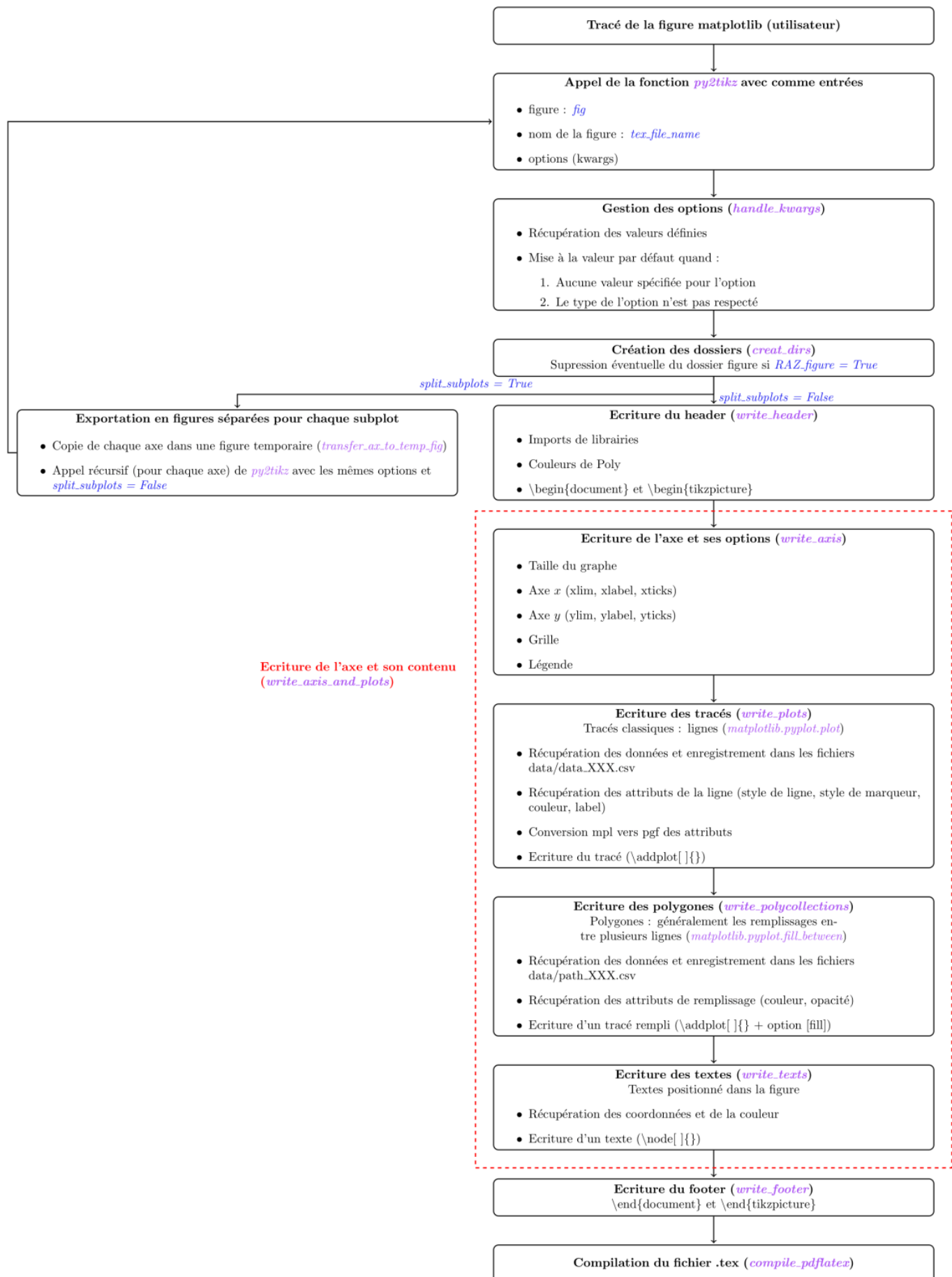
Le sous-dossier `Nom_du_fichier_tex` contient :

- Le fichier .tex contenant le code tikz
- Le fichier .pdf compilé, s'il n'est pas présent c'est qu'il y a une erreur dans le code et la compilation n'a pas fonctionné.
- Le dossier data qui contient tous les fichiers .tsv contenant les données du tracé.

## Démarche de génération d'une figure (inutile pour utilisateur)

La démarche utilisée est récapitulée dans la figure suivante :





Flowchart [pdf](#) de py2tikz

## Base de non régression

Afin de maintenir le code et de ne pas introduire de bugs avec de nouvelles fonctionnalités.

Cette base est dans le dépôt git du code et **doit être lancée avant chaque push sur le dépôt**. Il suffit d'exécuter le script qui va exécuter tous les exemples (pour l'instant 2) :

```
intelpyt3 && sh ./validation_non_regression.sh
```

Il faut ensuite s'assurer que les figures sont bien conformes et qu'elles ont été compilées correctement.

## Exemples

### Graphique 2D simple (markers)

La figure utilisée pour la base non régression est la suivante :

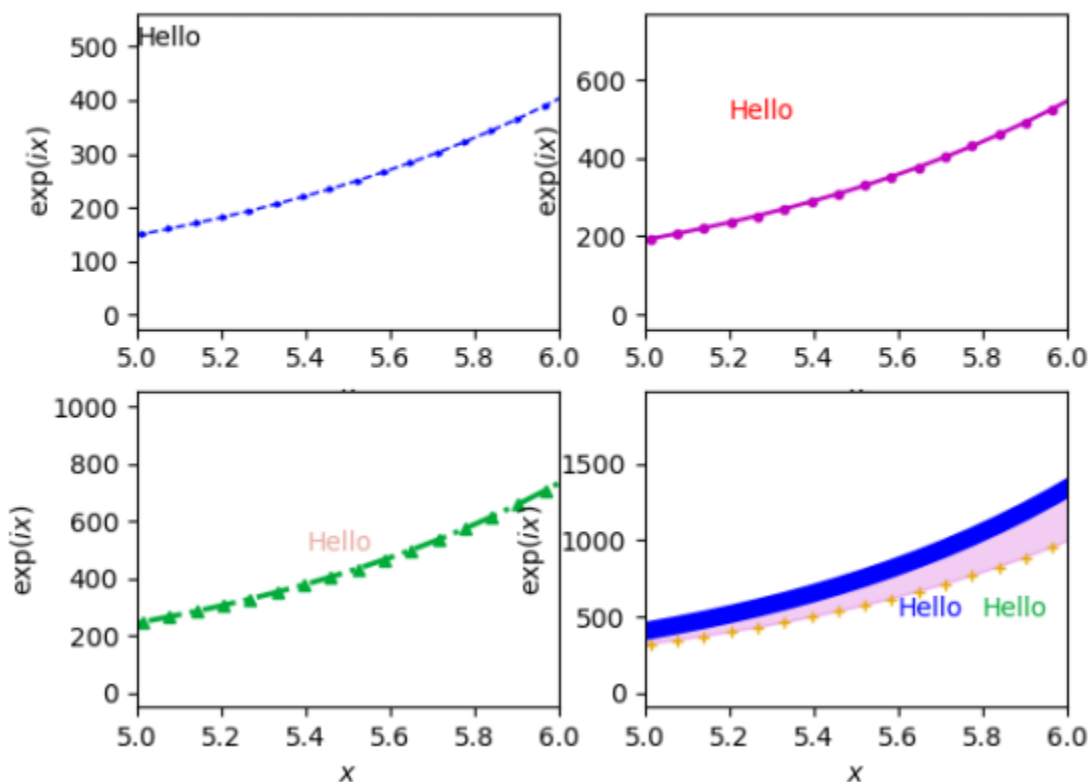


Figure matplotlib 2D simple

Résultats compilés de l'exécution de py2tikz sur cette figure :

- Exécution sans options (par défaut) : [pdf](#)
- Exécution avec options : [pdf](#)
- Exécution avec `split_subplots = True` et axes log : [pdf](#)

Document issu de la page wiki:

<https://lava-wiki.meca.polymtl.ca/ressources/latex/py2tikz/accueil?rev=1649702019>

Dernière mise à jour: **2023/04/05 08:59**