

Formation interne à Python

Matlab Vs Python

Premières ondulations pour les scientifiques

Séminaire 1

Julien Villemejeane

PRAG Institut d'Optique
LEnSE

<https://bit.ly/3LZV9uE>



Formation à Python

Contenus et objectifs des séminaires

Série de 5 séminaires (45min) / ateliers (1h)

MatLab vs Python

1

Premières ondulations pour les scientifiques

- Découverte du langage Python par l'exemple (en s'appuyant sur MatLab - Jeu des différences)
- Découverte de quelques bibliothèques utiles
- Découverte de l'environnement JupyterHub
- + Installation d'un environnement local

Deep Learning / Machine Learning

2

(Re)Découvrir les neurones informatiques

- Création d'un neurone et d'un réseau
- Découverte de Tensorflow

Traitement du signal

3

Etude de systèmes et de signaux

- Systèmes asservis / Fonction de transfert et représentation d'état (*via Scipy.signal et Control*)

Interfaçage

4

Développement d'une IHM simple

- Découverte de QT et Tkinter
- + Interfaçage avec une liaison série

Traitement d'images OpenCV

5

Maltraitance d'images avec OpenCV

- Découverte de la bibliothèque OpenCV



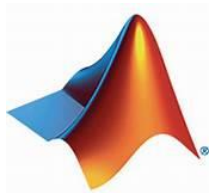
Formation à Python

Contenus et objectifs des séminaires

Série de 5 séminaires (45min) / ateliers (1h)

1	MatLab vs Python <i>Premières ondulations pour les scientifiques</i> <ul style="list-style-type: none">- Découverte du langage Python par l'exemple (en s'appuyant sur MatLab - Jeu des différences)- Découverte de quelques bibliothèques utiles- Découverte de l'environnement JupyterHub+ Installation d'un environnement local
2	Deep Learning / Machine Learning <i>(Re)Découvrir les neurones informatiques</i> <ul style="list-style-type: none">- Création d'un neurone et d'un réseau- Découverte de Tensorflow

3	Traitement du signal <i>Etude de systèmes et de signaux</i> <ul style="list-style-type: none">- Systèmes asservis / Fonction de transfert et représentation d'état (<i>via Scipy.signal et Control</i>)
4	Interfaçage <i>Développement d'une IHM simple</i> <ul style="list-style-type: none">- Découverte de QT et Tkinter+ Interfaçage avec une liaison série
5	Traitement d'images OpenCV <i>Maltraitance d'images avec OpenCV</i> <ul style="list-style-type: none">- Découverte de la bibliothèque OpenCV



MatLab vs Python

Objectifs « pédagogiques »

1

MatLab vs Python

Premières ondulations pour les scientifiques

- Découverte du langage Python par l'exemple (en s'appuyant sur MatLab - Jeu des différences)
 - Découverte de quelques bibliothèques utiles
 - Découverte de l'environnement JupyterHub
- + Installation d'un environnement local



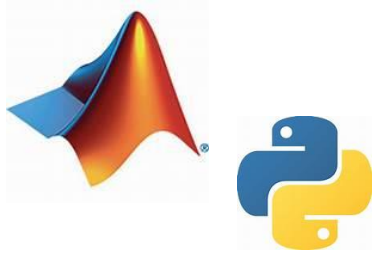
A la suite de ce séminaire / atelier, vous serez capable de :

- Ecrire et exécuter un script sous JupyterHub
- Utiliser des variables sous Python
- Créer et utiliser des vecteurs et des matrices
- Afficher des courbes 2D
- Calculer et afficher la FFT d'un signal 1D
- Ouvrir une image et l'afficher
- Calculer et afficher la FFT d'une image

Bibliothèques utiles

- Numpy
- Matplotlib / Pyplot
- PIL
- Scipy / signal













MatLab vs Python

Pourquoi Python ?

TIOBE 05/2022

May 2022	May 2021	Change	Programming Language		Ratings	Change	
1	2	⬆		Python	12.74%	+0.86%	
2	1	⬇		C	11.59%	-1.80%	
3	3			Java	10.99%	-0.74%	
4	4			C++	8.83%	+1.01%	
5	5			C#	6.39%	+1.98%	
...							
18	37	⬆		Lua	0.98%	+0.64%	
19	11	⬇		Ruby	0.86%	-0.64%	
20	15	⬇		MATLAB	0.82%	-0.41%	5



MatLab vs Python

Pourquoi Python ?

Top Programming Languages 2021

Choose a Ranking: **IEEE Spectrum** | Trending

Language Types: Web | Enterprise | Mobile | Embedded

Create Custom Ranking (Click to hide)

Language Ranking: **IEEE Spectrum**

Rank	Language	Type	Score
1	Python	Web, Enterprise, Embedded	100.0
2	C	Mobile, Enterprise, Embedded	94.7
3	C++	Mobile, Enterprise, Embedded	92.4
4	C#	Web, Mobile, Enterprise, Embedded	82.4
5	Arduino	Embedded	68.4
6	Rust	Web, Enterprise, Embedded	63.1
7	Assembly	Embedded	62.8
8	Verilog	Embedded	40.3

Choose a Ranking: **IEEE Spectrum** | Trending

Language Types: Web | Enterprise | Mobile | Embedded

Create Custom Ranking (Click to hide)

Language Ranking: **IEEE Spectrum**

Rank	Language	Type	Score
1	Python	Web, Enterprise, Embedded	100.0
2	Java	Web, Mobile, Enterprise	95.4
3	C	Mobile, Enterprise, Embedded	94.7
4	C++	Mobile, Enterprise, Embedded	92.4
5	C#	Web, Mobile, Enterprise, Embedded	82.4
6	R	Enterprise	81.7
7	Go	Web, Enterprise	77.7
8	Swift	Mobile, Enterprise	70.4

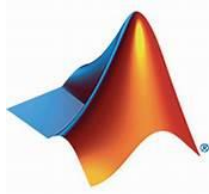
Choose a Ranking: **Jobs** | Trending

Language Types: Web | Enterprise | Mobile | Embedded

Create Custom Ranking (Click to hide)

Language Ranking: **Jobs**

Rank	Language	Type	Score
1	Python	Web, Enterprise, Embedded	100.0
2	C	Mobile, Enterprise, Embedded	96.0
3	Java	Web, Mobile, Enterprise	95.9
4	C++	Mobile, Enterprise, Embedded	88.3
5	Go	Web, Enterprise	87.3
6	R	Enterprise	85.7
7	C#	Web, Mobile, Enterprise, Embedded	79.8
8	SQL	Enterprise	71.9



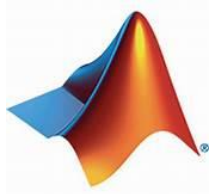
MatLab vs Python

Pourquoi Python ?

MAIS ??

Table 4. Normalized global results for Energy, Time, and Memory

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

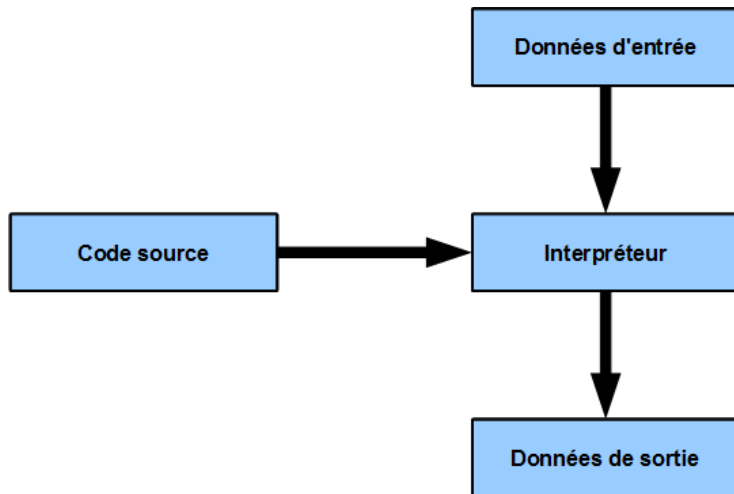


MatLab vs Python

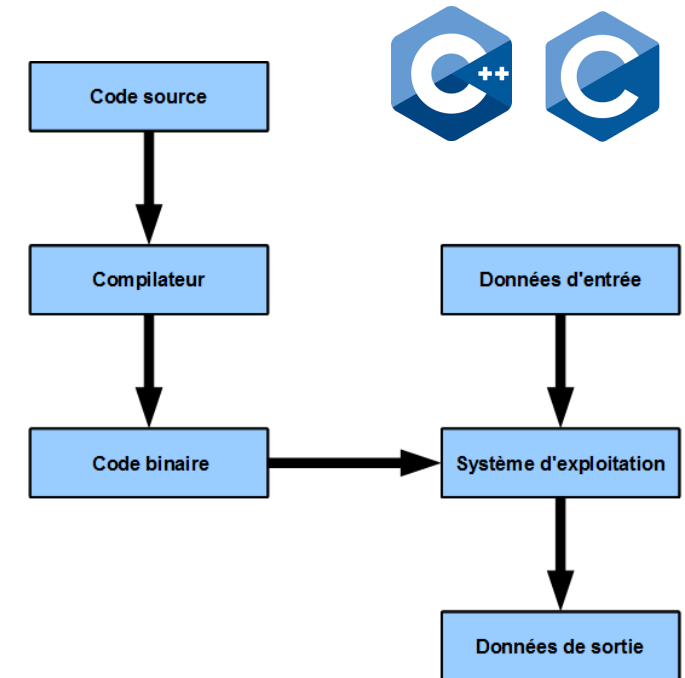
Python : langage interprété



Python : un langage interprété



≠/≠ langage compilé



Langage orienté objet

Utilisation « procédurale » possible



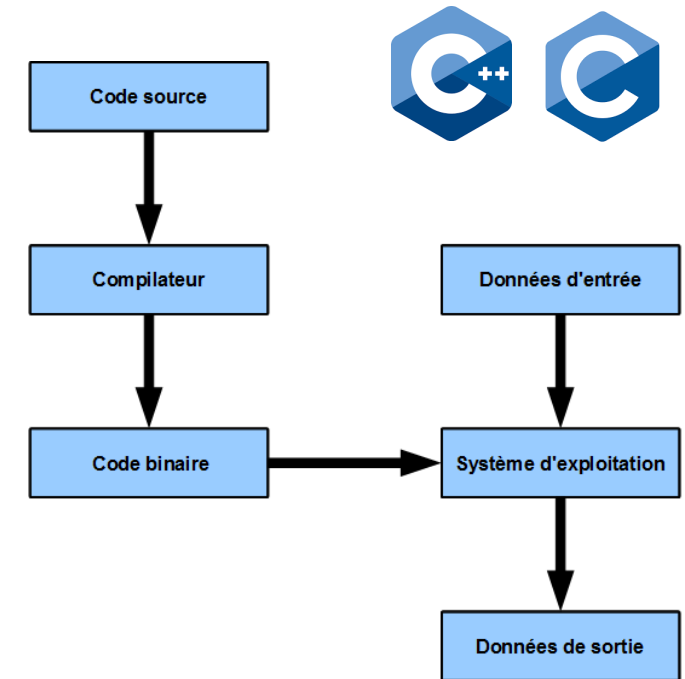
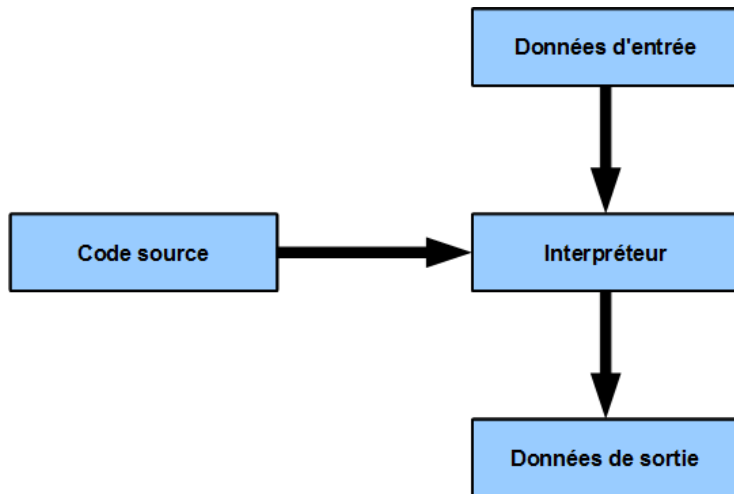
MatLab vs Python

Python : langage interprété



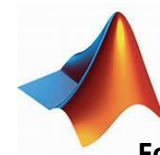
Python : un langage interprété

≠ langage compilé

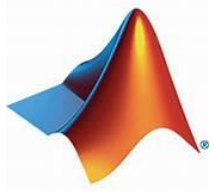


Langage orienté objet

Utilisation « procédurale » possible



Fonctions optimisées



MatLab vs Python

Environnements de travail

Distributions et Environnements

Distribution : ensemble de logiciels et de librairies
incluant des environnements et des interpréteurs

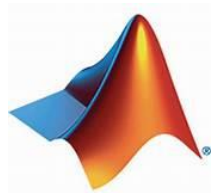


Environnement (IDE) : ensemble d'outils pour l'édition et
l'interprétation des commandes / programmes
incluant des interpréteurs et des éditeurs de texte



Bibliothèques : ensemble de modules
supplémentaires
incluant des classes, des fonctions...



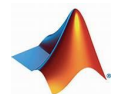


MatLab vs Python

Environnements de travail



Distributions et Environnements



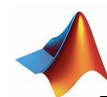
Distribution : ensemble de logiciels et de librairies
incluant des environnements et des interpréteurs



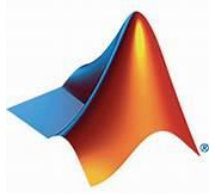
Environnement (IDE) : ensemble d'outils pour l'édition et
l'interprétation des commandes / programmes
incluant des interpréteurs et des éditeurs de texte



Bibliothèques : ensemble de modules
supplémentaires
incluant des classes, des fonctions...



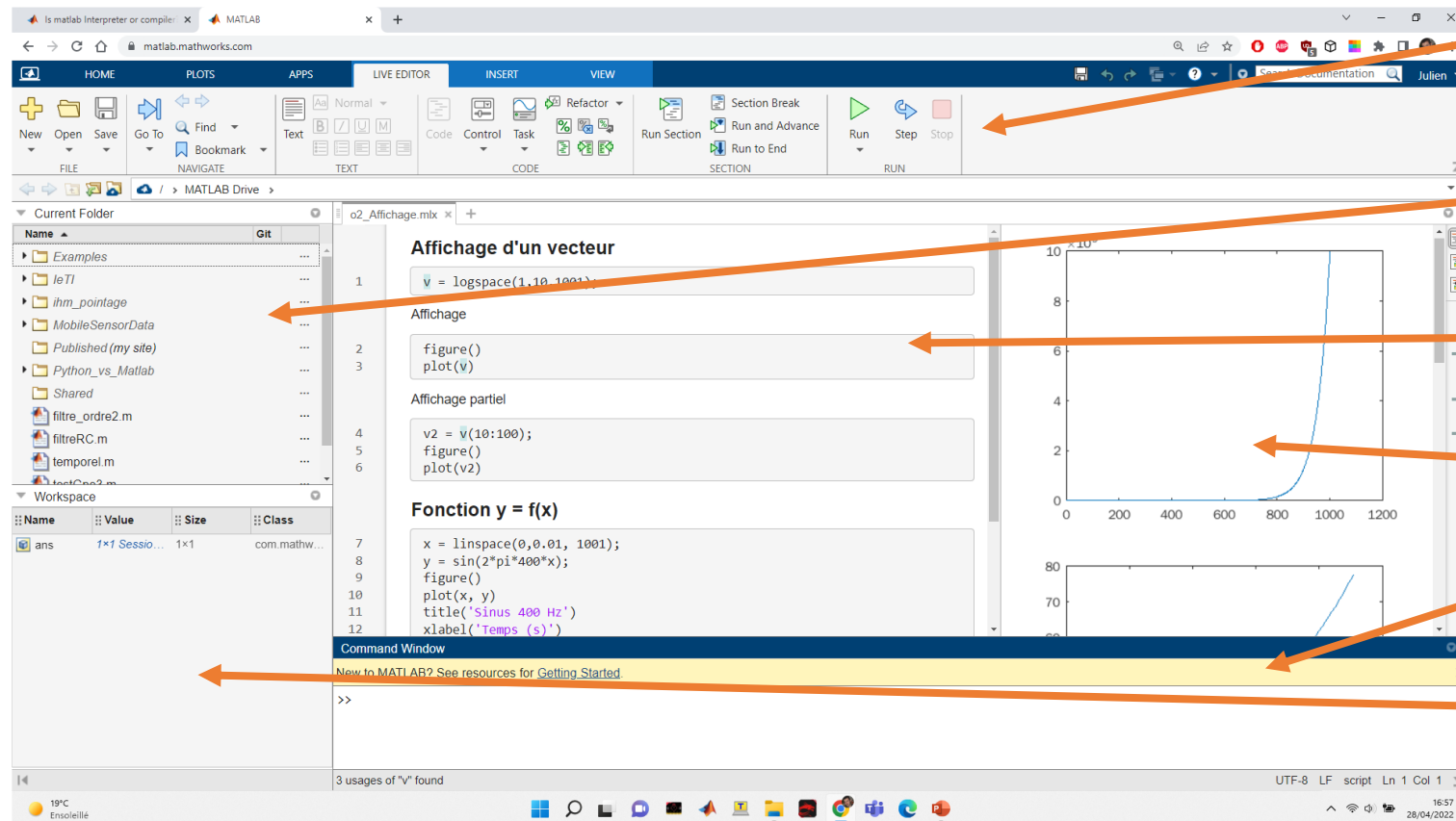
Boîtes à outils



MatLab vs Python

Environnement MatLab

MatLab (version Online + LiveScript)



Outils

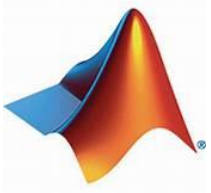
Espace de travail

Editeur de texte
(Scripts ou Live Scripts)

Affichage des résultats

Commandes

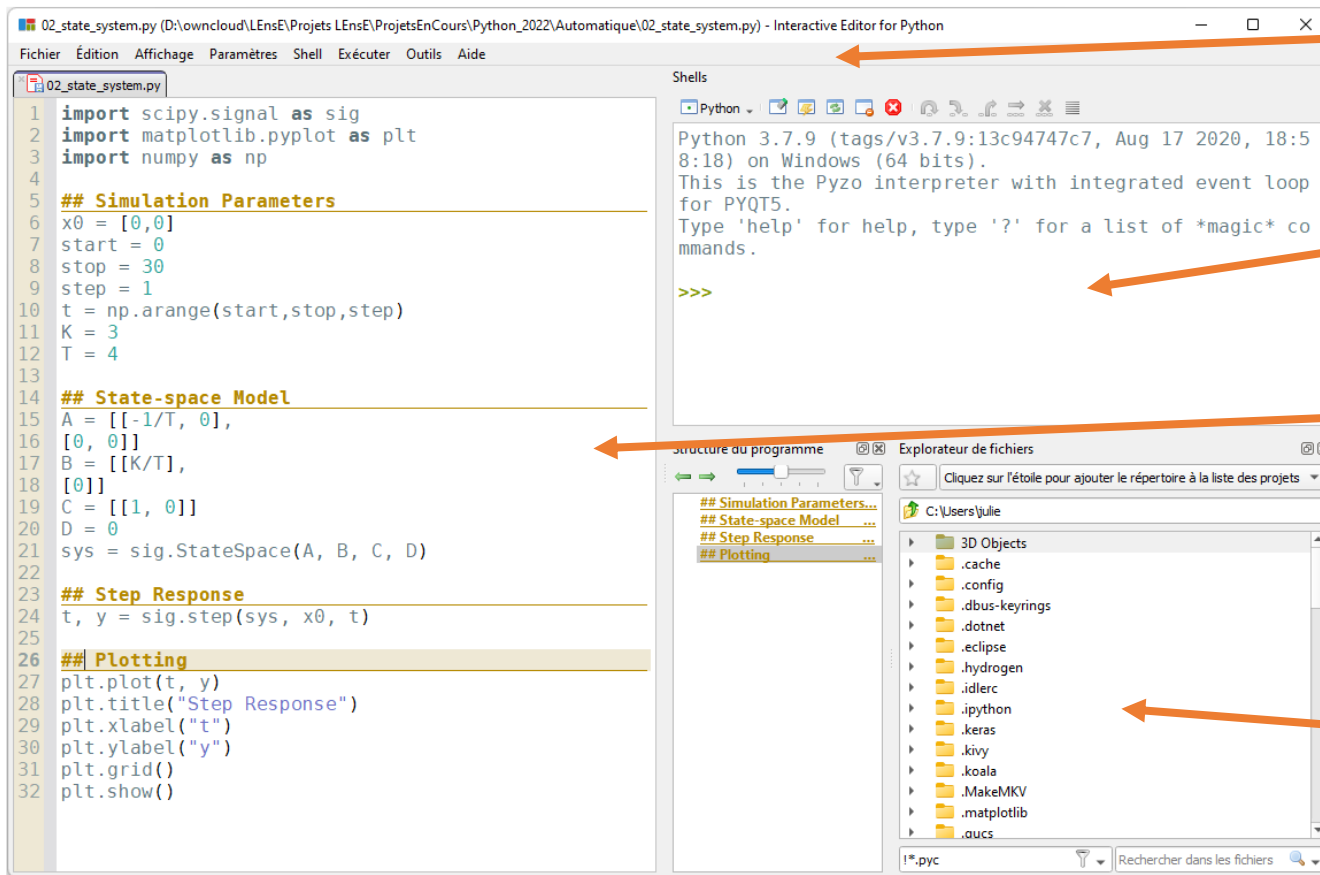
Variables



MatLab vs Python

Environnement MatLab

Pyzo (interpréteur local)

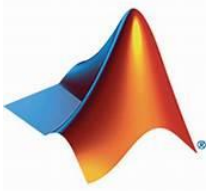


Outils

Commandes

Editeur de texte
(Scripts ou Live Scripts)

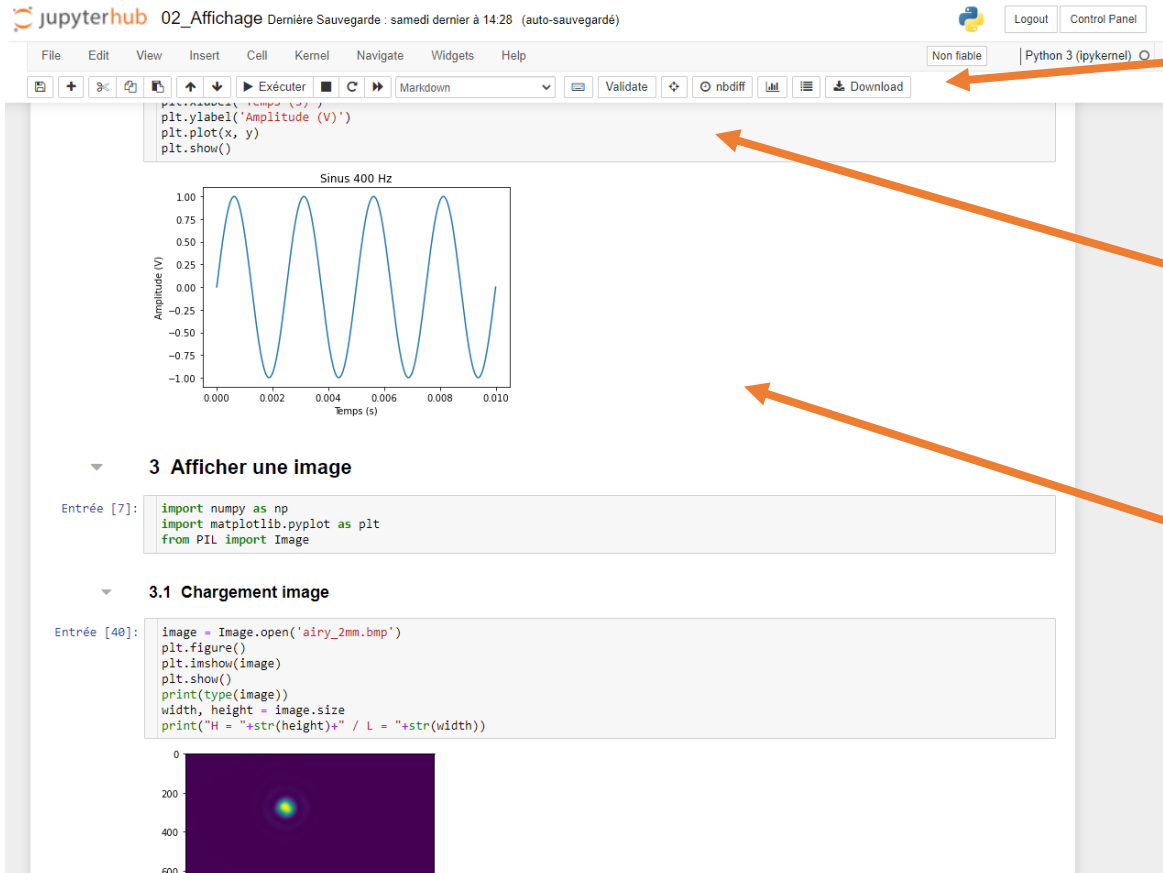
Espace de travail



MatLab vs Python

Environnement MatLab

Jupyter Hub (*interpréteur en ligne*) == *Live Script MatLab*

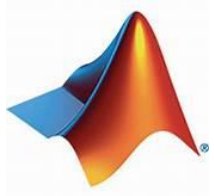


Outils

Editeur de texte
(Scripts ou Live Scripts)

Affichage des résultats

<https://jupyterhub.ijclab.in2p3.fr/>



MatLab vs Python

Premières lignes



Variables

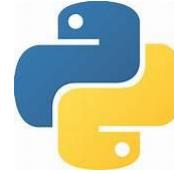
```
a = 2 + 3
```

```
a = 5
```



```
a = 2 + 3
```

```
—
```





MatLab vs Python

Premières lignes

Variables

```
a = 2 + 3
```

```
a = 5
```

```
a = 2 + 3
```

```
print( a )
```

```
print( 'a =', a )
```

```
5
```

```
a = 5
```

BILAN

Variables : ==

Affichage valeurs : ~~



MatLab vs Python

Premières lignes

Vecteurs

```
b = [1, 2, 3]
```

```
b = 1x3
```

```
1
```

```
2
```

```
3
```

```
b(1)
```

```
ans = 1
```

```
b = [1, 2, 3]  
print( b )
```

BILAN

Variables : ==

Affichage valeurs : ~~



MatLab vs Python

Premières lignes

Vecteurs

```
b = [1, 2, 3]
```

```
b = 1x3
```

```
1
```

```
2
```

```
3
```

```
b(1)
```

```
ans = 1
```

```
b = [1, 2, 3]
```

```
print( b )
```

```
[1, 2, 3]
```

```
print( b[1] )
```

BILAN

Variables : ==

Affichage valeurs : ~~



MatLab vs Python

Premières lignes

Vecteurs

```
b = [1, 2, 3]
```

```
b = 1x3
```

```
1
```

```
2
```

```
3
```

```
b(1)
```

```
ans = 1
```

```
b = [1, 2, 3]
```

```
print( b )
```

```
[1, 2, 3]
```

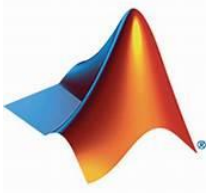
```
print( b[1] )
```

```
2
```

BILAN

Vecteurs : ==

MAIS indices allant de 0 à N-1



MatLab vs Python

Premières lignes

Vecteurs

```
b = [1, 2, 3]
```

```
b = 1x3
```

```
1
```

```
2
```

```
3
```

```
b(1)
```

```
ans = 1
```

```
b = [1, 2, 3]  
print( b )
```

```
[1, 2, 3]
```

```
print( b[1] )
```

```
2
```

```
print( type( b ) )
```

```
<class 'list' >
```

BILAN

Vecteurs : ==

MAIS indices allant de 0 à N-1



MatLab vs Python

Premières lignes

Matrices : déclaration

```
m = [1,2,3 ; 4,5,6]
```

b = 2x3

1	2	3
4	5	6

```
m = [[1,2,3],[4,5,6]]  
print( m )  
print( type( m ))
```



MatLab vs Python

Premières lignes

Matrices : déclaration

```
m = [1,2,3 ; 4,5,6]
```

b = 2x3

1	2	3
4	5	6

```
m = [[1,2,3],[4,5,6]]
```

```
print( m )
```

```
print( type( m ))
```

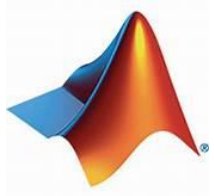
```
[[1, 2, 3], [4, 5, 6]]
```

```
<class 'list'>
```

BILAN

Matrices : ==

MAIS indices allant de 0 à N-1



MatLab vs Python

Premières lignes

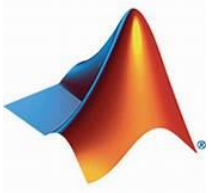
Matrices : somme

```
m = [1,2,3 ; 4,5,6] ;  
m2 = [1,2,3 ; 4,5,6] ;  
msum = m + m2
```

msum = 2x3

2	4	6
8	10	12

```
m = [[1,2,3],[4,5,6]]  
m2 = [[1,2,3],[4,5,6]]  
msum = m + m2  
print( msum )
```



MatLab vs Python

Premières lignes

Matrices : somme

```
m = [1,2,3 ; 4,5,6] ;  
m2 = [1,2,3 ; 4,5,6] ;  
msum = m + m2
```

msum = 2x3

2	4	6
8	10	12

```
m = [[1,2,3],[4,5,6]]  
m2 = [[1,2,3],[4,5,6]]  
msum = m + m2  
print( msum )
```

```
[[1, 2, 3], [4, 5, 6], [1, 2, 3], [4, 5, 6]]
```

BILAN

Matrices : \neq

Python n'est pas un logiciel de calculs matriciels !!





MatLab vs Python

Premières lignes

Matrices : somme

```
m = [1,2,3 ; 4,5,6] ;  
m2 = [1,2,3 ; 4,5,6] ;  
msum = m + m2
```

msum = 2x3

2	4	6
8	10	12

```
m = [[1,2,3],[4,5,6]]  
m2 = [[1,2,3],[4,5,6]]  
msum = m + m2  
print( msum )
```

```
[[1, 2, 3], [4, 5, 6], [1, 2, 3], [4, 5, 6]]
```

BILAN

Matrices : \neq

Python n'est pas un logiciel de calculs matriciels !!





MatLab vs Python

Première bibliothèque utile

Vecteurs et matrices : utilisation de la bibliothèque NUMPY



```
m = [1,2,3 ; 4,5,6] ;  
m2 = [1,2,3 ; 4,5,6] ;  
msum = m + m2
```

```
msum = 2x3  
      2      4      6  
      8     10     12
```

```
import numpy as np  
ma = np.array( [1, 2, 3] )  
print( ma )
```

```
[1 2 3]
```

```
print( type( ma ) )
```

```
<class 'numpy.ndarray'>
```



MatLab vs Python

Première bibliothèque utile

Utilisation d'une bibliothèque

```
import numpy  
ma = numpy.array( [1, 2, 3] )
```

```
import numpy as np  
ma = np.array( [1, 2, 3] )
```

```
from matplotlib import pyplot  
pyplot.figure()
```

```
from matplotlib import pyplot as plt  
plt.figure()
```

```
import numpy as np  
ma = np.array( [1, 2, 3] )  
print( ma )
```

```
[1 2 3]
```

```
print( type( ma ) )
```

```
<class 'numpy.ndarray'>
```



MatLab vs Python

Calculs matriciels

Vecteurs et matrices : somme



```
m = [1,2,3 ; 4,5,6] ;  
m2 = [1,2,3 ; 4,5,6] ;  
msum = m + m2
```

```
msum = 2x3  
      2      4      6  
      8     10     12
```

```
import numpy as np  
mb = np.array( [[1,2,3] , [4,5,6]] )  
mc = np.array( [[1,2,3] , [4,5,6]] )  
mm = mb + mc  
print( mm )
```

```
[[ 2  4  6]  
 [ 8 10 12]]
```



MatLab vs Python

Calculs matriciels

Variables : nombres complexes



```
mk = [1j , 2 , 3] ;
```

mk = 1x3 complex

0 + 1i 2 + 0i 3 + 0i

```
import numpy as np  
mk = np.array([1j, 2, 3], dtype=complex)  
print( mk )
```

```
[ 0+1j  2+0j  3+0j]
```

```
nk = 1j + 3  
print( nk )  
print( type( nk ) )
```

```
(1j + 3)  
<class 'complex'>
```



MatLab vs Python

Calculs matriciels

Matrices : produits termes à termes

```
m1 = [1,2,3 ; 4,5,6] ;  
m2 = [1,2,3 ; 4,5,6] ;  
ms = m1 .* m2
```

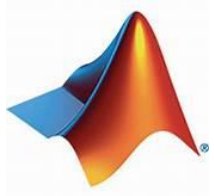
ms =

1	4	9
16	25	36

```
import numpy as np  
m1 = np.array([[1,2,3],[4,5,6]])  
m2 = np.array([[1,2,3],[4,5,6]])  
ms = m1 * m2  
print( ms )
```

[[1	4	9]
[16	25	36]]





MatLab vs Python

Calculs matriciels

Matrices : produits matriciels



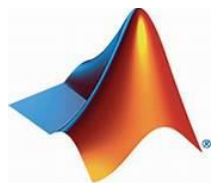
```
m1 = [1,2,3 ; 4,5,6] ;  
m2 = [1,2,3 ; 4,5,6] ;  
ms = m1 * m2'
```

ms = 2x2

14	32
32	77

```
import numpy as np  
m1 = np.array([[1,2,3],[4,5,6]])  
m2 = np.array([[1,2,3],[4,5,6]])  
ms = np.dot( m1 , m2.T )  
print( ms )
```

[[14	32]
[32	77]]



MatLab vs Python

Autre fonctions

Interaction avec l'utilisateur



```
k = input('Saisir une valeur :')
k
```

```
k = input( 'Saisir une valeur :' )
print( k )
```

Lecture d'un fichier de données (txt)

1	1, 3
2	2, 7
3	3, 8
4	4, 2
5	5, 4

```
from numpy import loadtxt

lines = loadtxt("data.txt", delimiter=",")
print( type( lines ) )
print( lines.shape )
```

```
for line in lines:
    print(line)
```

```
<class 'numpy.ndarray'>
(5, 2)
```

```
[1 3]
[2 7] ...
```




MatLab vs Python

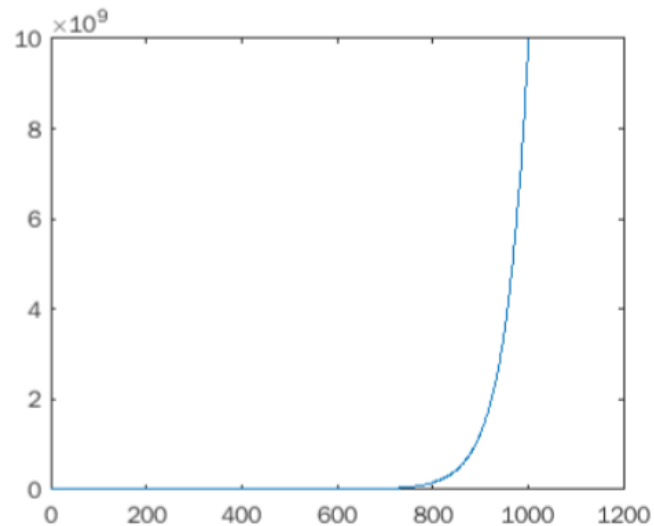
Affichages graphiques



Affichage



```
v = logspace( 1, 10, 1001 );  
figure()  
plot( v )
```



```
import numpy as np  
v = np.logspace( 1, 10, 1001 )
```



MatLab vs Python

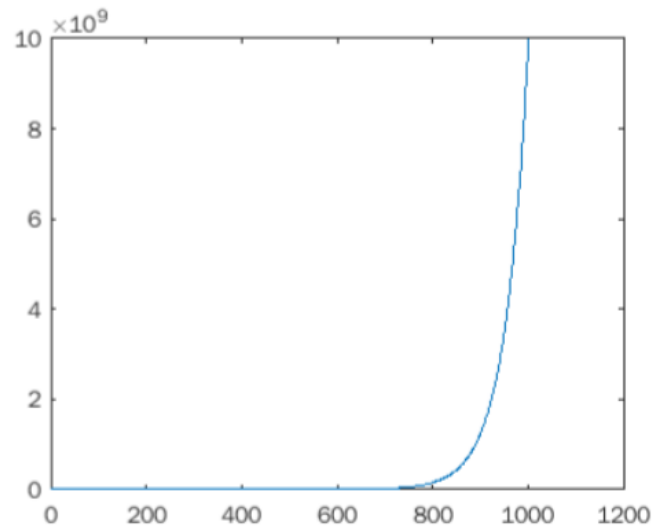
Affichages graphiques



Affichage



```
v = logspace( 1, 10, 1001 );  
figure()  
plot( v )
```



```
import numpy as np  
v = np.logspace( 1, 10, 1001 )
```





MatLab vs Python

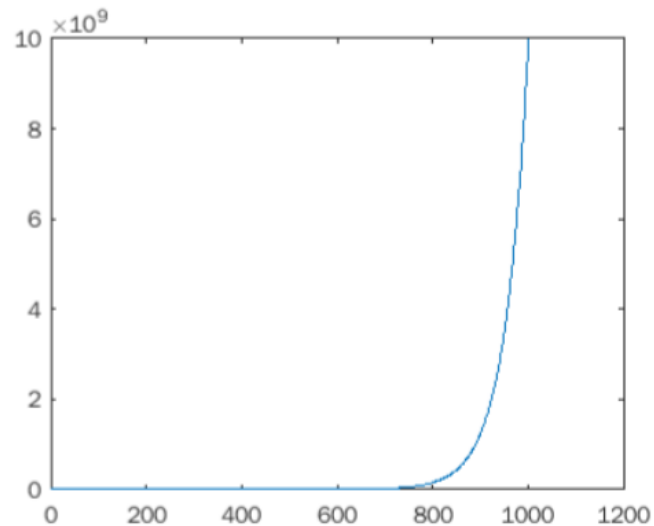
Affichages graphiques



Affichage



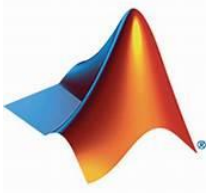
```
v = logspace( 1, 10, 1001 );  
figure()  
plot( v )
```



```
import numpy as np  
v = np.logspace( 1, 10, 1001 )
```

```
import matplotlib.pyplot as plt  
plt.figure()  
plt.plot( v )  
plt.show()
```

matplotlib



MatLab vs Python

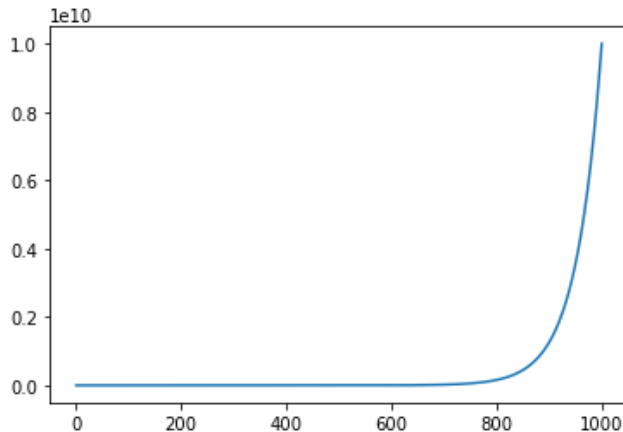
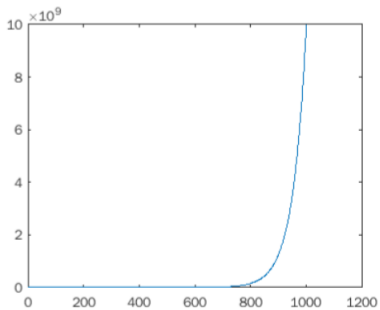
Affichages graphiques



Affichage



```
v = logspace( 1, 10, 1001 );  
figure()  
plot( v )
```



```
import numpy as np  
v = np.logspace( 1, 10, 1001 )
```

```
import matplotlib.pyplot as plt  
plt.figure()  
plt.plot( v )  
plt.show()
```

matplotlib



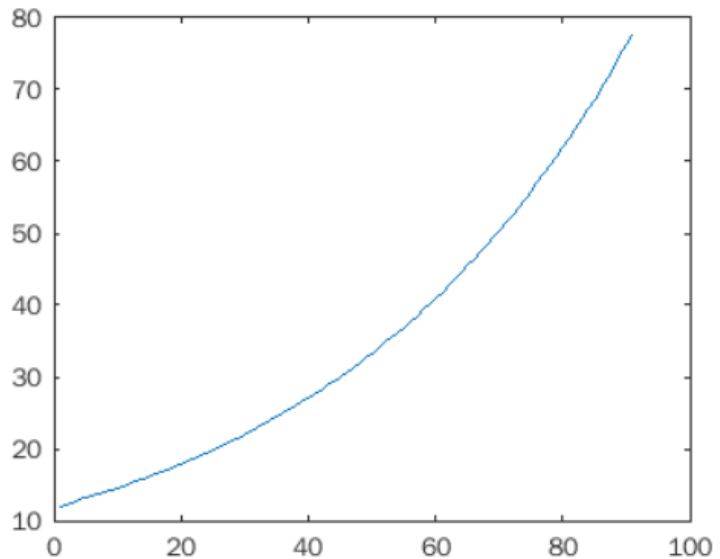
MatLab vs Python

Affichages graphiques

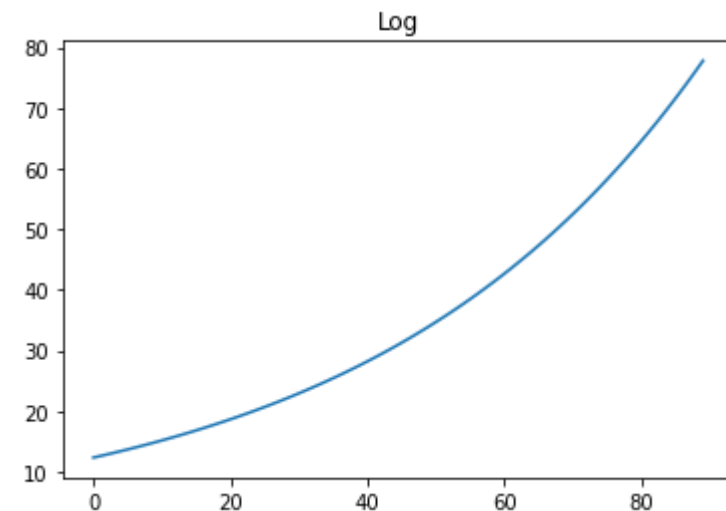


Affichage partiel

```
v2 = v(10:100);  
figure()  
plot( v2 )
```



```
v2 = v[ 10 : 100 ]  
plt.figure()  
plt.title( 'Log' )  
plt.plot( v2 )  
plt.show()
```





MatLab vs Python

Affichages graphiques

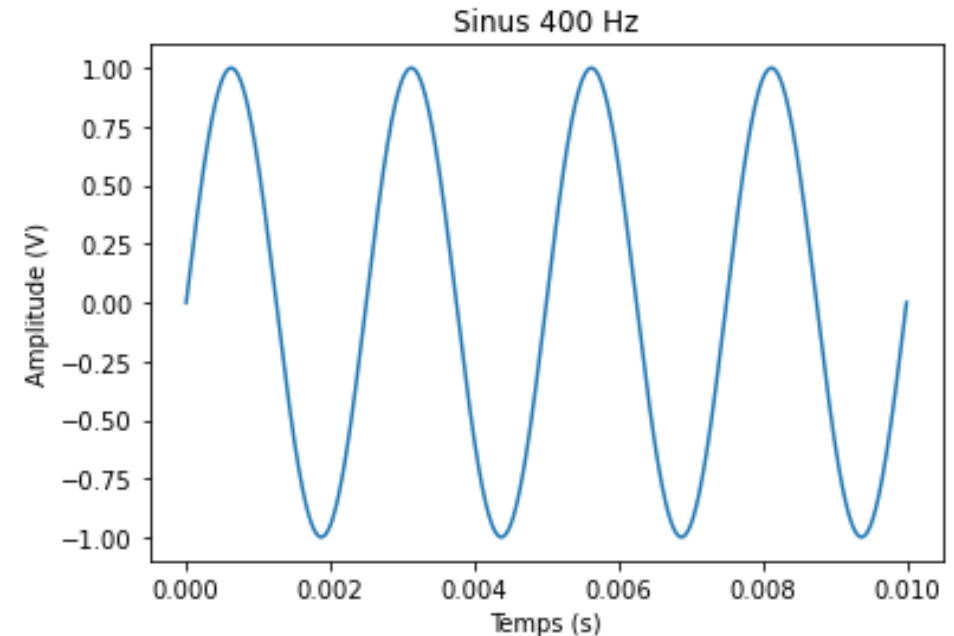


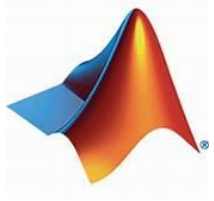
Courbe $y = f(x)$

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0,0.01,1001)
y = np.sin(2*np.pi*400*x)
plt.figure()
plt.title( 'Sinus 400 Hz' )
plt.xlabel( 'Temps (s)' )
plt.ylabel( 'Amplitude (V)' )
plt.plot( x , y )
plt.show()
```

matplotlib





MatLab vs Python

Images



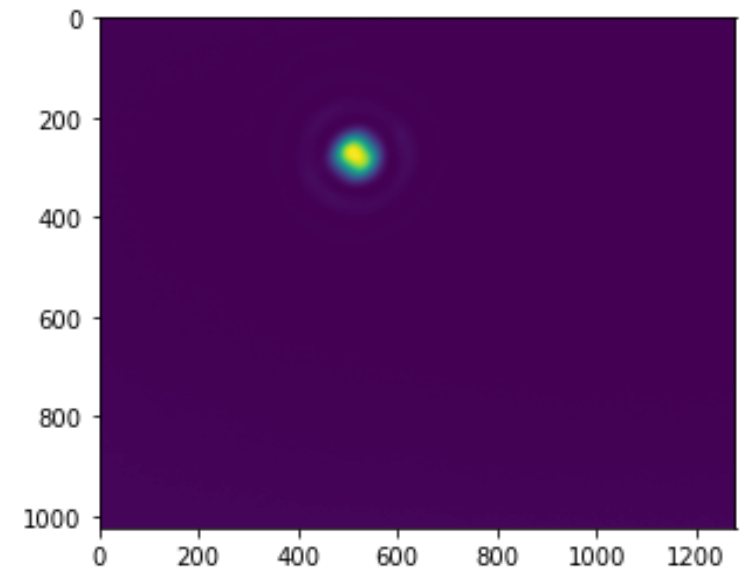
Afficher une image

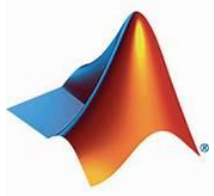
```
import matplotlib.pyplot as plt
from PIL import Image

image = Image.open( 'airy_2mm.bmp' )
plt.figure()
plt.imshow( image )
plt.show()
print( type( image ) )
width, height = image.size
print( "H = " + str( height ) + " / L = " + str(width))
```

```
<class'PIL.BmpImagePlugin.BmpImageFile'>
H = 1024 / L = 1280
```

matplotlib





MatLab vs Python

Images



Afficher une image / coupe horizontale

```
import numpy as np
```

```
image_data = np.array( image )  
print( type( image_data ) )  
print( image_data.shape )
```

```
<class 'numpy.ndarray'>  
(1024, 1280)
```

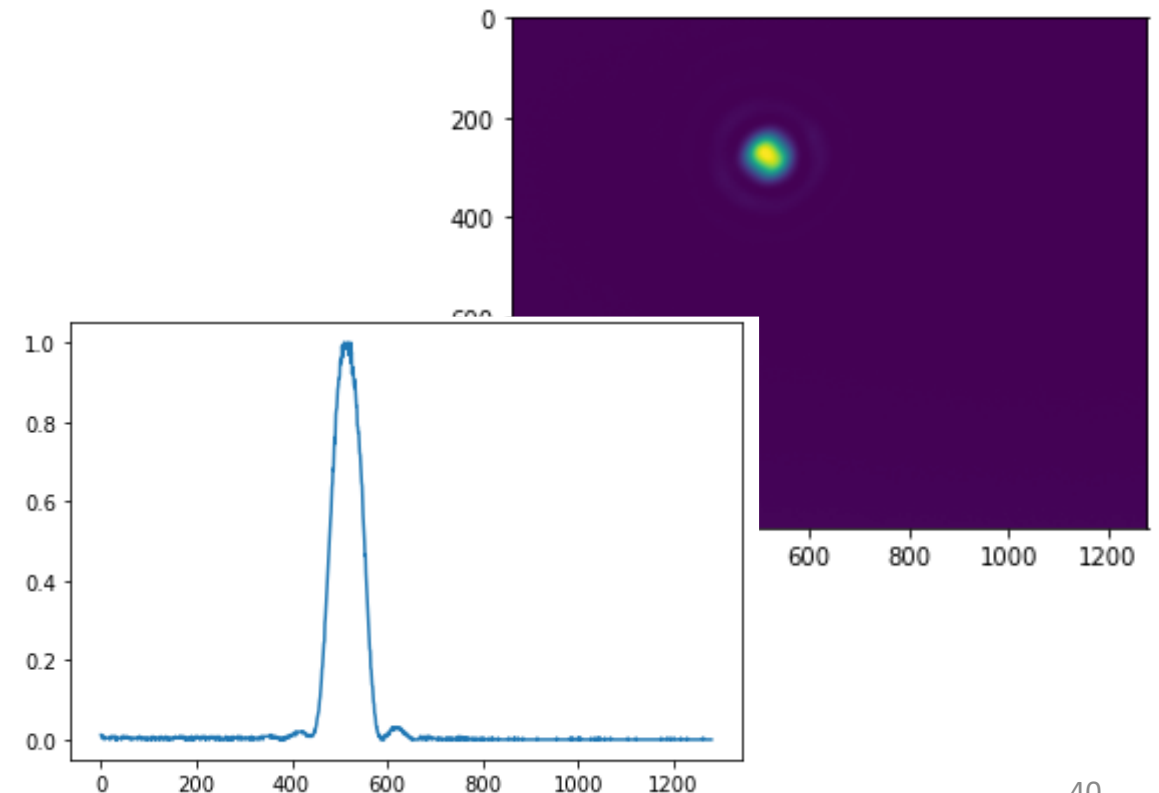
```
coupe_hor = image_data[ 280 , : ]  
plt.figure()  
plt.plot(coupe_hor/255)  
plt.show()
```



pillow

matplotlib

NumPy





MatLab vs Python

Images

Afficher une image / maximum d'une matrice

```
max_image = np.argmax( image_data )
max_image_iH = np.floor( max_image % h_image )
max_image_iL = np.floor( max_image / h_image )
print( max_image )
print( "Ligne = " + str( int( max_image_iL )))
print( "Colonne = " + str( int( max_image_iH )))
```

H = 1024 / L = 1280

338430 (*max_image*)

Ligne = 330

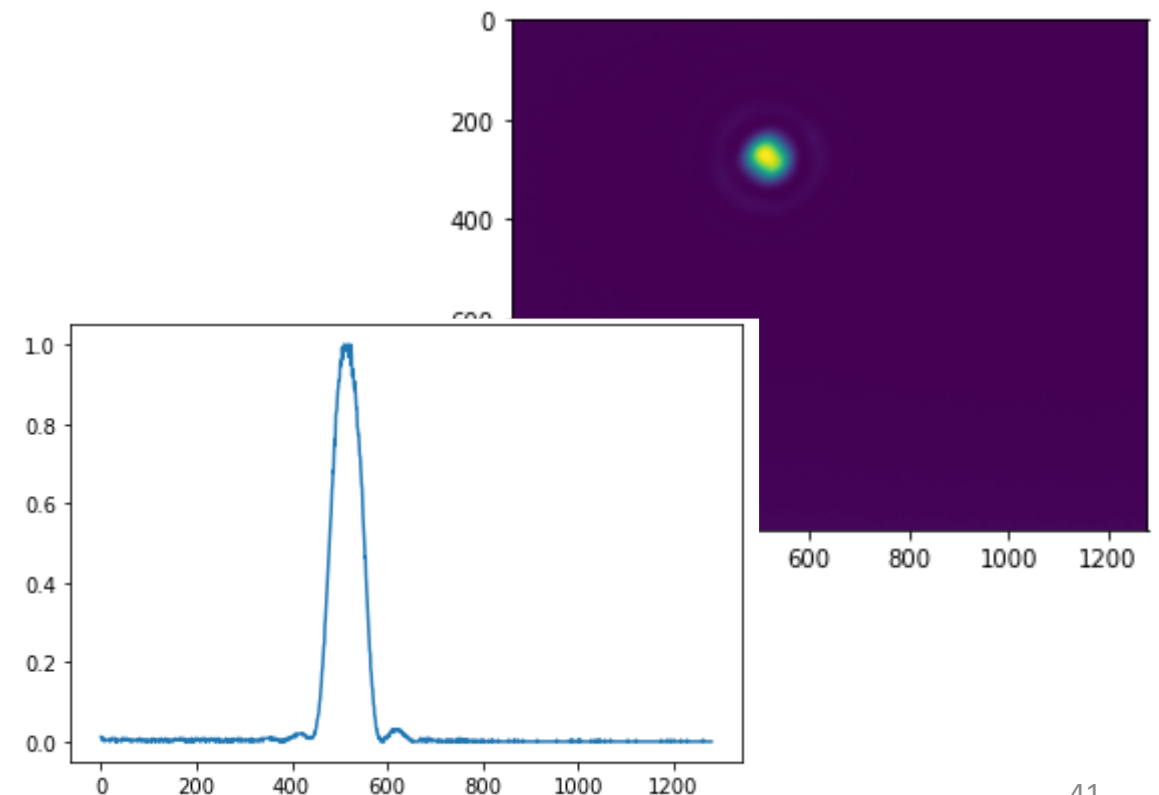
Colonne = 510



pillow

matplotlib

NumPy





MatLab vs Python

Traitement du signal

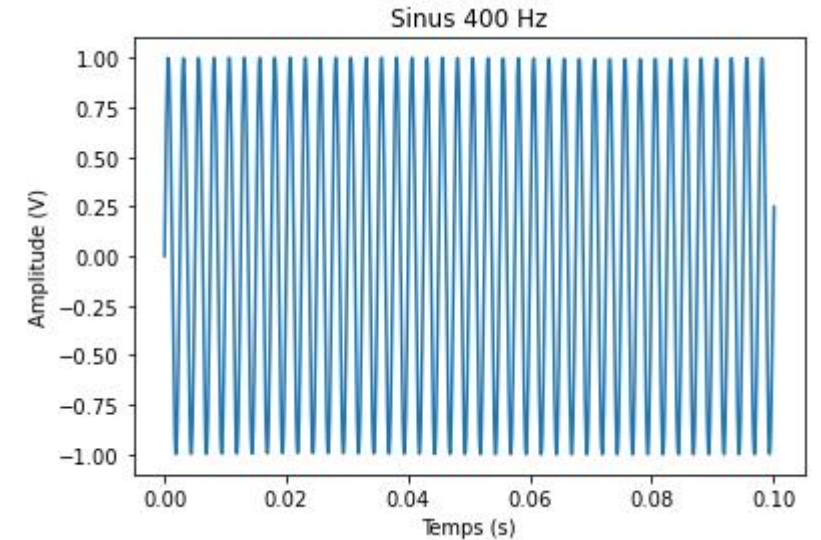


FFT 1D

```
Fe = 1e4  
Npoints = 1001  
T = Npoints / Fe  
t = np.linspace(0, T-T/Npoints, Npoints)  
v = np.sin(2*np.pi*400*t)  
plt.figure()  
plt.title('Sinus 400 Hz')  
plt.xlabel('Temps (s)')  
plt.ylabel('Amplitude (V)')  
plt.plot(t, v)  
plt.show()
```

matplotlib

NumPy





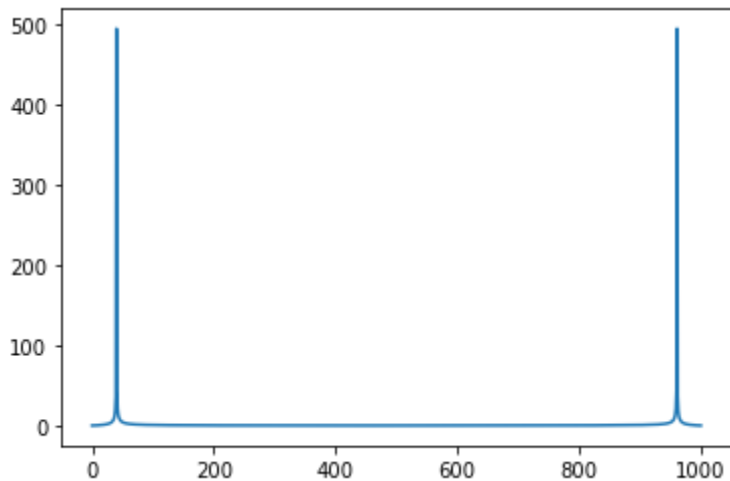
MatLab vs Python

Traitement du signal



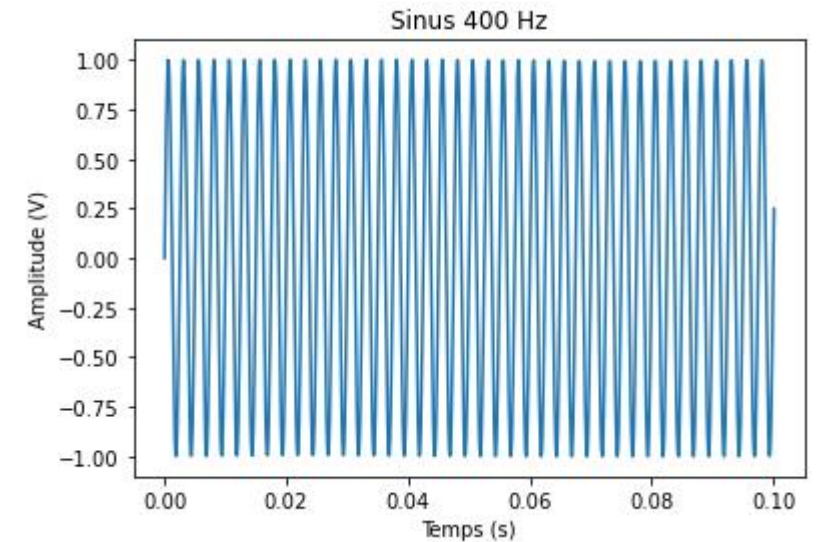
FFT 1D

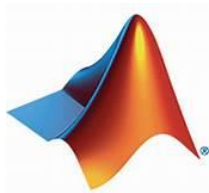
```
TFv = np.fft.fft( v )  
plt.figure()  
plt.plot( np.absolute( TFv ))  
plt.show()
```



matplotlib

NumPy





MatLab vs Python

Traitement du signal

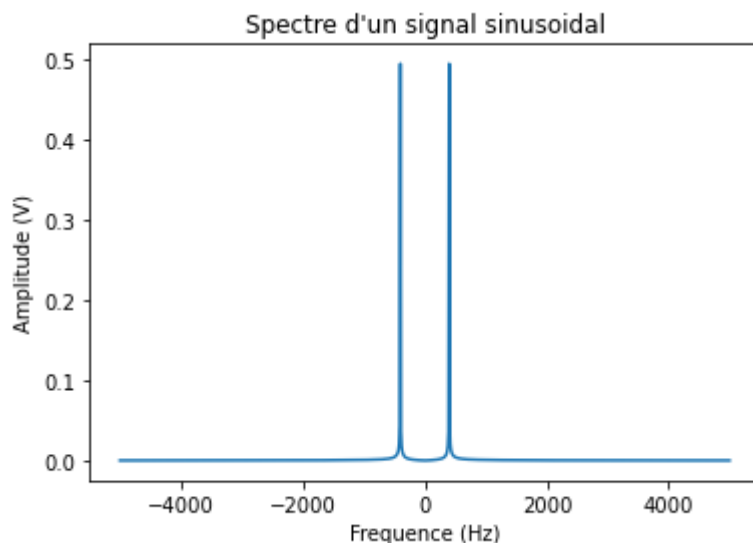
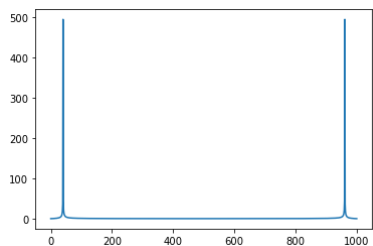


FFT 1D

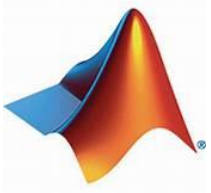
matplotlib

NumPy

```
TFv = np.fft.fft( v )  
plt.figure()  
plt.plot( np.absolute( TFv ))  
plt.show()
```



```
freq = np.linspace( -Fe/2+Fe/(2*Npoints), Fe/2-  
Fe/(2*Npoints), Npoints ) # pour  
Npoints IMPAIR  
plt.figure()  
TFvs = np.fft.fftshift(np.absolute(TFv))/Npoints  
plt.plot( freq , TFvs )  
plt.show()
```



MatLab vs Python

Traitement d'images

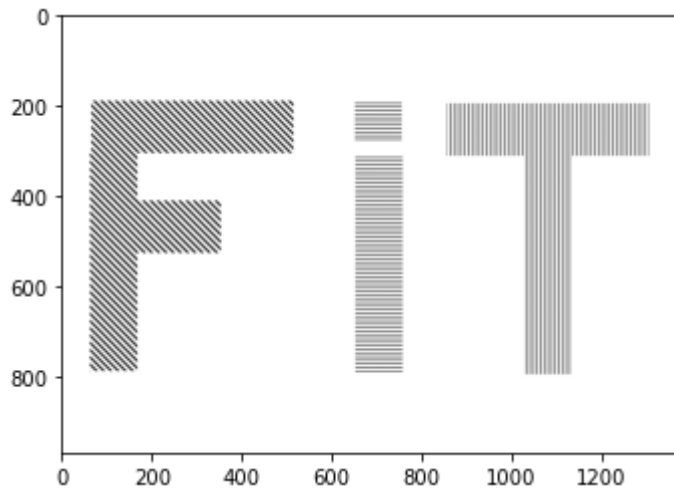


FFT 2D



pillow

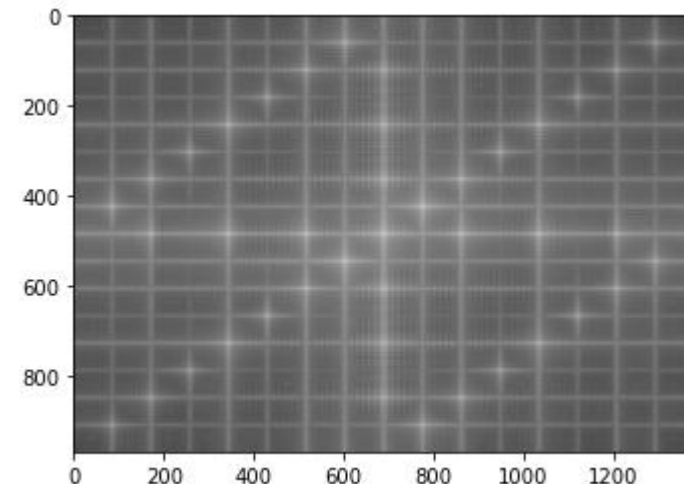
matplotlib



```
TFimage = np.fft.fft2( image_data )  
mTFimage = np.absolute( TFimage )
```

```
TFimage_shift = np.fft.fftshift( TFimage )  
mTFimage_shift = np.absolute(TFimage_shift)
```

```
plt.figure()  
plt.imshow( np.log(mTFimage_shift), cmap='gray')  
plt.show()
```





MatLab vs Python

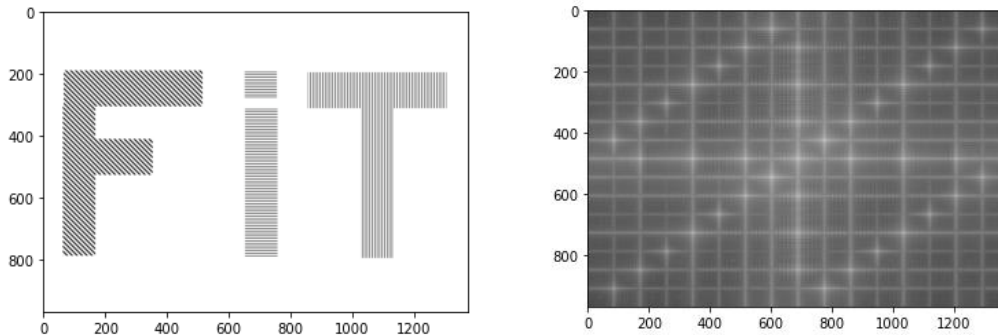
Traitement d'images

FFT 2D / Traitement et FFT-1

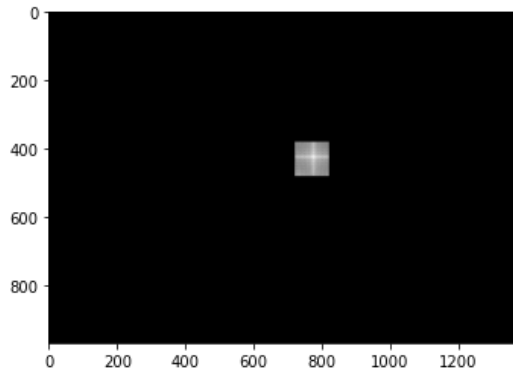


pillow

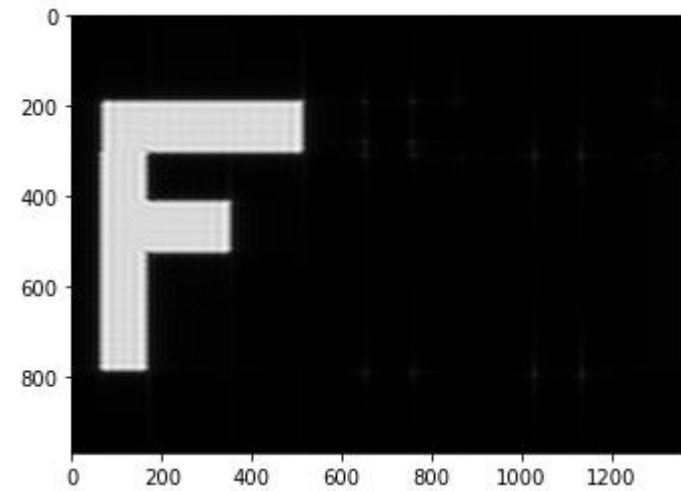
matplotlib

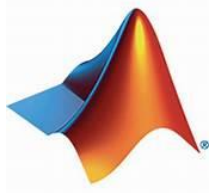


```
new_tf_shift = np.zeros((h_image, l_image), dtype='complex') + 1  
new_tf_shift[380:480, 720:820] = TFimage_shift[380:480, 720:820]
```



```
new_tf = np.fft.fftshift( new_tf_shift )  
new_image = np.fft.ifft2( new_tf )
```





MatLab vs Python

Etude de systèmes



Systèmes par fonction de transfert

```
import scipy.signal as sig
import matplotlib.pyplot as plt
import numpy as np
```

```
num1 = np.array([3])
num2 = np.array([2, 1])
num = np.convolve(num1, num2)
den1 = np.array([3, 1])
den2 = np.array([5, 1])
den = np.convolve(den1, den2)
H = sig.TransferFunction(num, den)
print('H(s) =', H)
```



```
H(s) = TransferFunctionContinuous(
array([0.4, 0.2]),
array([1.      , 0.53333333, 0.06666667]),
dt: None
)
```

matplotlib

NumPy