**eBook Gratuit**

# APPRENEZ scipy

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#scipy

# Table des matières


À propos ... 1

Chapitre 1: Commencer avec Scipy ... 2

Remarques ... 2

Versions ... 2

Examples ... 4

Installation ou configuration ... 4

Convertir une matrice éparse en une matrice dense en utilisant SciPy ... 5

Des versions ... 5

Manipulation d'image à l'aide de Scipy (Redimensionnement d'image de base) ... 5

Basic Hello World ... 6

Chapitre 2: Comment écrire une fonction Jacobian pour optimiser.minimize ... 8

Syntaxe ... 8

Remarques ... 8

Examples ... 8

Exemple d'optimisation (en or) ... 8

Exemple d'optimisation (Brent) ... 9

Fonction Rosenbrock ... 10

Chapitre 3: Fonctions d'ajustement avec scipy.optimize curve_fit ... 12

Introduction ... 12

Examples ... 12

Adapter une fonction aux données d'un histogramme ... 12

Chapitre 4: Lisser un signal ... 15

Examples ... 15

Utilisation d'un filtre Savitzky – Golay ... 15

Chapitre 5: rv_continuous pour la distribution avec les paramètres ... 17

Examples ... 17

Binôme négatif sur les réels positifs ... 17

Crédits ... 18

# À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: scipy

It is an unofficial and free scipy ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official scipy.



Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

# Chapitre 1: Commencer avec Scipy

## Remarques

À propos de Scipy

SciPy est un ensemble d'algorithmes mathématiques et de fonctions pratiques basés sur l'extension Numpy de Python. Il ajoute une puissance significative à la session Python interactive en fournissant à l'utilisateur des commandes et des classes de haut niveau pour manipuler et visualiser les données. Avec SciPy, une session Python interactive devient un environnement de traitement de données et de prototypage rivalisant avec des systèmes tels que MATLAB, IDL, Octave, R-Lab et SciLab.

L'avantage supplémentaire de baser SciPy sur Python est que cela rend également un langage de programmation puissant disponible pour le développement de programmes sophistiqués et d'applications spécialisées. Les applications scientifiques utilisant SciPy bénéficient du développement de modules supplémentaires dans de nombreux niches du paysage logiciel par des développeurs du monde entier. Tout, de la programmation parallèle aux sous-programmes et classes Web et de base de données, a été mis à la disposition du programmeur Python. Toute cette puissance est disponible en plus des bibliothèques mathématiques de SciPy.

## Versions

| Version | Date de sortie |
|---|---|
| 0.19.0 | 2017-03-09 |
| 0.18.0 | 2016-07-25 |
| 0,17,0 | 2016-01-22 |
| 0.16.1 | 2015-10-24 |
| 0,16,0 | 2015-07-23 |
| 0,16b2 | 2015-05-24 |
| 0,16b1 | 2015-05-12 |
| 0.15.1 | 2015-01-18 |
| 0,15,0 | 2015-01-11 |
| 0.14.1 | 2014-12-30 |
| 0.14.1rc1 | 2014-12-14 |

| Version | Date de sortie |
| --- | --- |
| 0,14,0 | 2014-05-03 |
| 0.14.0rc2 | 2014-04-23 |
| 0.14.0rc1 | 2014-04-02 |
| 0.14.0b1 | 2014-03-16 |
| 0.13.3 | 2014-02-04 |
| 0.13.2 | 2013-12-07 |
| 0.13.1 | 2013-11-16 |
| 0.13.0 | 2013-10-19 |
| 0.13.0rc1 | 2013-10-10 |
| 0.12.1 | 2013-10-08 |
| 0.12.0 | 2013-04-06 |
| 0.12.0rc1 | 2013-03-29 |
| 0.12.0b1 | 2013-02-16 |
| 0.11.0 | 2012-09-24 |
| 0.11.0rc2 | 2012-08-12 |
| 0.11.0rc1 | 2012-07-17 |
| 0.11.0b1 | 2012-06-12 |
| 0,10,1 | 2012-02-26 |
| 0.10.1rc2 | 2012-02-19 |
| 0.10.1rc1 | 2012-02-10 |
| 0.10.0 | 2011-11-13 |
| 0.10.0rc1 | 2011-11-03 |
| 0.10.0b2 | 2011-09-16 |
| 0.10.0b1 | 2011-09-11 |
| 0.9.0 | 2011-02-27 |

# Examples

## Installation ou configuration

Scipy contient des parties écrites en C, C ++ et Fortran qui doivent être compilées avant utilisation. Par conséquent, assurez-vous que les compilateurs et les en-têtes de développement Python nécessaires sont installés. Avoir du code compilé signifie également que Scipy a besoin d'étapes supplémentaires pour importer depuis des sources de développement, qui sont expliquées ci-dessous.

Placez une copie du dépôt Scipy principal dans Github sur votre propre compte, puis créez votre référentiel local via:

```
$ git clone git@github.com:YOURUSERNAME/scipy.git scipy
$ cd scipy
$ git remote add upstream git://github.com/scipy/scipy.git
```

Pour générer la version de développement de Scipy et exécuter des tests, générez des shells interactifs avec les chemins d'importation Python correctement configurés, etc. Effectuez l'une des actions suivantes:

```
$ python runtests.py -v
$ python runtests.py -v -s optimize
$ python runtests.py -v -t scipy/special/tests/test_basic.py:test_xlogy
$ python runtests.py --ipython
$ python runtests.py --python somescript.py
$ python runtests.py --bench
```

Cela construit d'abord Scipy, donc cela peut prendre un certain temps la première fois. Si vous spécifiez `-n` , les tests seront exécutés par rapport à la version de Scipy (le cas échéant) trouvée sur le PYTHONPATH actuel.

L'utilisation de runtests.py est l'approche recommandée pour exécuter des tests. Il existe également un certain nombre d'alternatives, par exemple la création sur place ou l'installation dans un environnement virtuel. Certains tests sont très lents et doivent être activés séparément.

Lien vers l'API

**Ubuntu et Debian**

Exécuter la commande

```
sudo apt-get install python-numpy python-scipy python-matplotlib ipython ipython-notebook
python-pandas python-sympy python-nose
```

Les versions d'Ubuntu 12.10 ou plus récent et Debian 7.0 ou plus récent répondent à la spécification de pile SciPy actuelle. Les utilisateurs peuvent également souhaiter ajouter le référentiel NeuroDebian pour les packages SciPy supplémentaires.

## Convertir une matrice éparse en une matrice dense en utilisant SciPy

```
from scipy.sparse import csr_matrix
A = csr_matrix([[1,0,2],[0,3,0]])
>>>A
<2x3 sparse matrix of type '<type 'numpy.int64'>'
   with 3 stored elements in Compressed Sparse Row format>
>>> A.todense()
  matrix([[1, 0, 2],
          [0, 3, 0]])
>>> A.toarray()
     array([[1, 0, 2],
            [0, 3, 0]])
```

## Des versions

La première version de SciPy, vsn 0.10, est sortie le 14 août 2001. La version actuelle de SciPy (correcte au 26 juillet 2016) est la version 0.17 (stable) avec la version v18 à venir. Les détails des anciennes versions sont listés ici

## Manipulation d'image à l'aide de Scipy (Redimensionnement d'image de base)

SciPy fournit des fonctions de base de manipulation d'images. Celles-ci incluent des fonctions pour lire des images du disque dans des tableaux numpy, pour écrire des tableaux numpy sur le disque en tant qu'images et pour redimensionner des images.

Dans le code suivant, une seule image est utilisée. Il est teinté, redimensionné et enregistré. Les images originales et résultantes sont présentées ci-dessous:

```
import numpy as np  //scipy is numpy-dependent

from scipy.misc import imread, imsave, imresize   //image resizing functions

# Read an JPEG image into a numpy array
img = imread('assets/cat.jpg')
print img.dtype, img.shape  # Prints "uint8 (400, 248, 3)"

# We can tint the image by scaling each of the color channels
# by a different scalar constant. The image has shape (400, 248, 3);
# we multiply it by the array [1, 0.95, 0.9] of shape (3,);
# numpy broadcasting means that this leaves the red channel unchanged,
# and multiplies the green and blue channels by 0.95 and 0.9
# respectively.
img_tinted = img * [1, 0.95, 0.9]

# Resize the tinted image to be 300 by 300 pixels.
img_tinted = imresize(img_tinted, (300, 300))

# Write the tinted image back to disk
imsave('assets/cat_tinted.jpg', img_tinted)
```

Référence

## Basic Hello World

Créez un fichier (par exemple hello_world.py) dans un éditeur de texte ou un éditeur python si vous en avez un installé ( choisissez-en un si vous ne le faites pas - SublimeText, Eclipse, NetBeans, SciTe ... il y en a beaucoup!)

```
hwld = 'Hello world'
print(hwld)
```

Notez que les variables python n'ont pas besoin d'être déclarées explicitement. la déclaration se produit lorsque vous affectez une valeur avec le signe égal (=) à une variable.

La sortie des deux lignes de code ci-dessus indique que la chaîne "Hello World" sera affichée.

Les fonctions écrites en Python peuvent également être utilisées dans iPython.

Dans ce cas, vous pouvez utiliser le fichier enregistré 'hello_world.py' dans IPython comme suit:

```
In [1]: %run hello_world.py
#run file to get output below
Hello world
In [2]: wld
#show what value of wld var is
Out[2]: 'Hello world'
In [3]: %whowld
#display info on variable wld (name/type/value)

Variable     Type     Data/Info
```

```
----------------------------
wld         str     Hello world
```

Si vous le souhaitez, vous pouvez utiliser deux variables, par exemple une pour hello et une pour world et les concaténer en utilisant le signe plus (+):

```
h = 'Hello '
w = "world!'
print(h+w)

#this will also output Hello World, only this time with an exclamation mark..
```

Lire Commencer avec Scipy en ligne: https://riptutorial.com/fr/scipy/topic/2128/commencer-avec-scipy

# Chapitre 2: Comment écrire une fonction Jacobian pour optimiser.minimize

## Syntaxe

1. importer numpy comme np
2. à partir de scipy.optimize import _minimize
3. de scipy import spécial
4. importez matplotlib.pyplot en tant que plt

## Remarques

Notez le trait de soulignement avant "minimiser" lors de l'importation depuis scipy.optimize; '_minimize' En outre, j'ai testé les fonctions de ce lien avant de faire cette section, et j'ai constaté que j'avais moins de problèmes / cela fonctionnait plus rapidement si j'importais séparément 'special'. La fonction Rosenbrock sur la page liée était incorrecte - vous devez d'abord configurer la barre de couleur; J'ai posté un autre code mais je pense que ça pourrait être mieux.

D'autres exemples à venir

Voir ici pour une explication de la matrice de Hesse

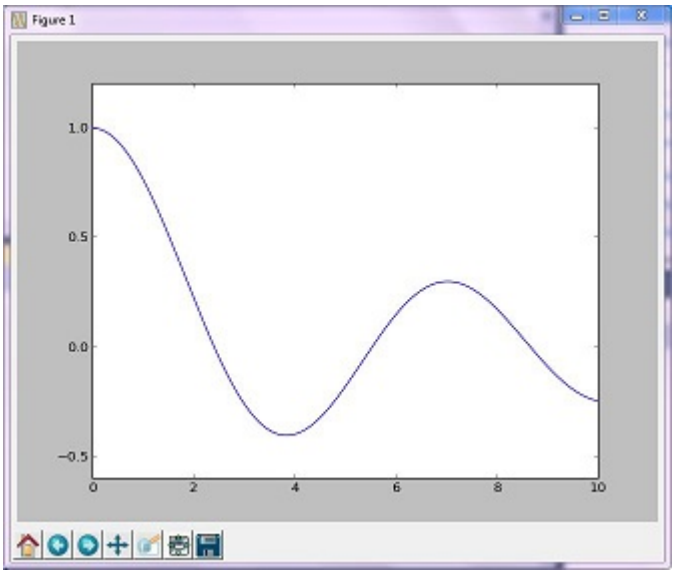
## Examples

### Exemple d'optimisation (en or)

La méthode 'Golden' minimise une fonction unimodale en réduisant la plage des valeurs extrêmes

```
import numpy as np
from scipy.optimize import _minimize
from scipy import special
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 500)
y = special.j0(x)
optimize.minimize_scalar(special.j0, method='golden')
plt.plot(x, y)
plt.show()
```

Image résultante

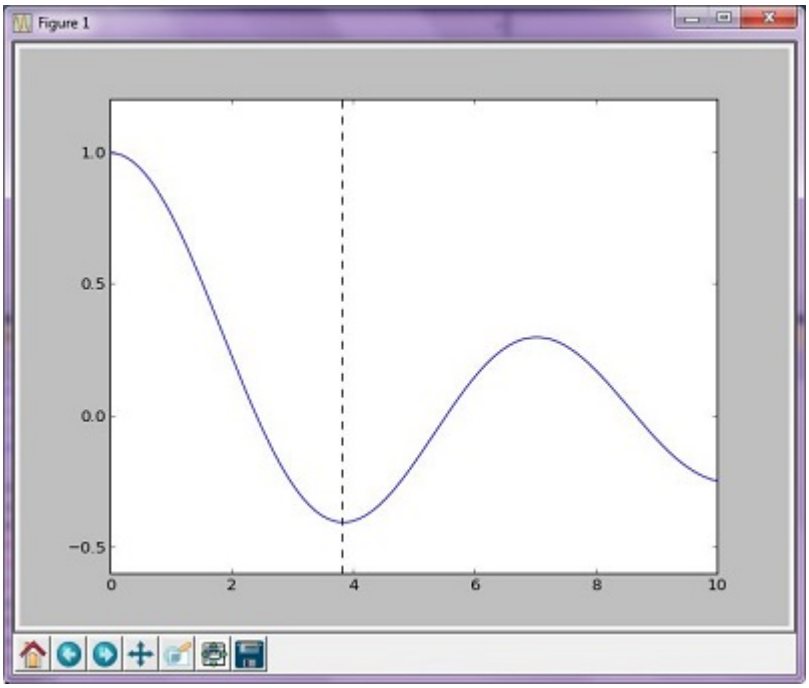



## Exemple d'optimisation (Brent)

La méthode de Brent est une combinaison d'algorithmes plus complexe d'autres algorithmes de recherche de racines; Cependant, le graphique résultant n'est pas très différent du graphique généré par la méthode golden.

```
import numpy as np
import scipy.optimize as opt
from scipy import special
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 500)
y = special.j0(x)
# j0 is the Bessel function of 1st kind, 0th order
minimize_result = opt.minimize_scalar(special.j0, method='brent')
the_answer = minimize_result['x']
minimized_value = minimize_result['fun']
# Note: minimize_result is a dictionary with several fields describing the optimizer,
# whether it was successful, etc. The value of x that gives us our minimum is accessed
# with the key 'x'. The value of j0 at that x value is accessed with the key 'fun'.
plt.plot(x, y)
plt.axvline(the_answer, linestyle='--', color='k')
plt.show()
print("The function's minimum occurs at x = {0} and y = {1}".format(the_answer,
minimized_value))
```

Graphique résultant



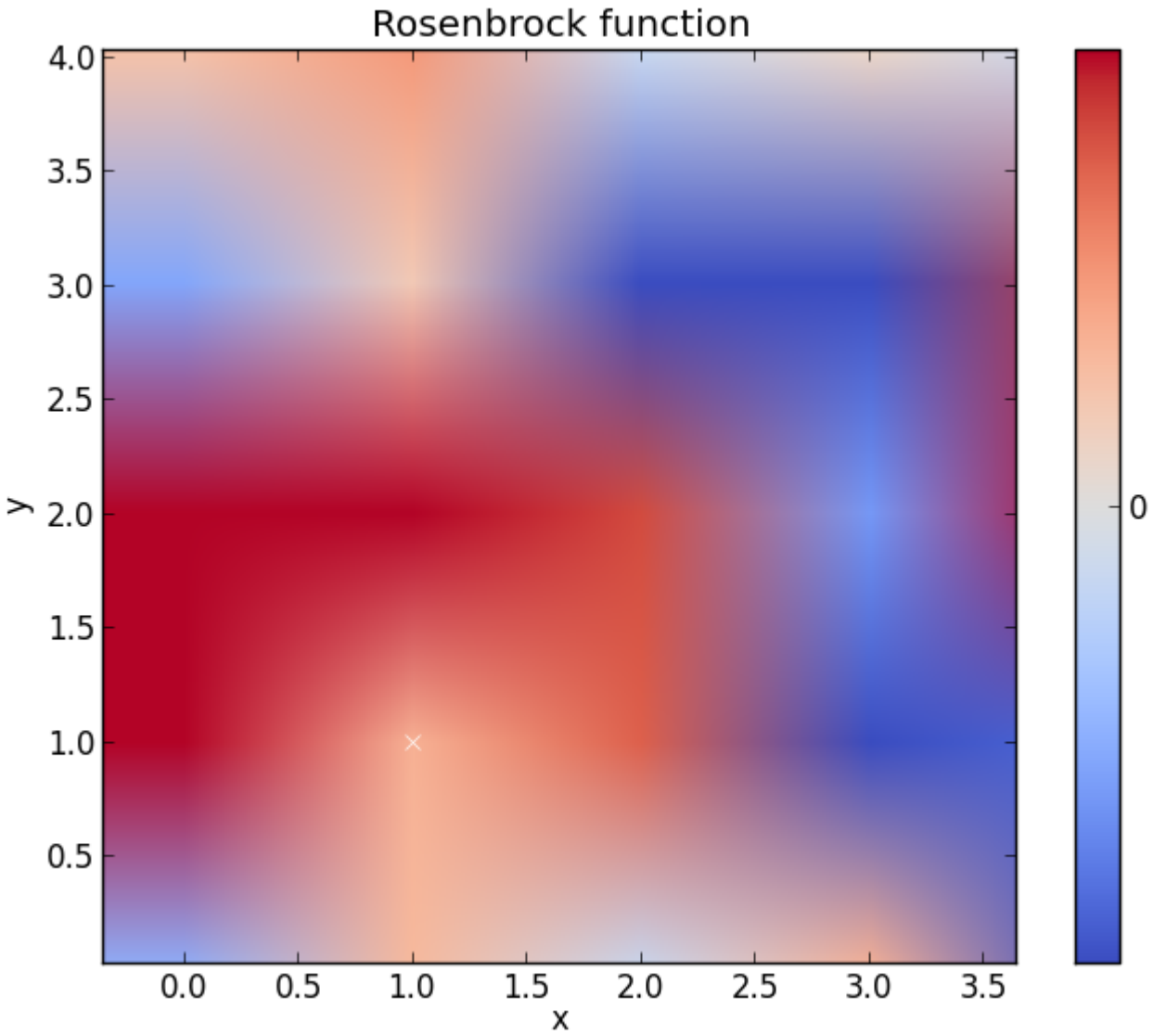

Les sorties:

```
The function's minimum occurs at x = 3.8317059554863437 and y = -0.4027593957025531
```

## Fonction Rosenbrock

Pense que cela pourrait par exemple être mieux, mais vous obtenez l'essentiel

```
import numpy as np
from scipy.optimize import _minimize
from scipy import special
import matplotlib.pyplot as plt
from matplotlib import cm
from numpy.random import randn

x, y = np.mgrid[-2:2:100j, -2:2:100j]
plt.pcolor(x, y, optimize.rosen([x, y]))
plt.plot(1, 1, 'xw')

# Make plot with vertical (default) colorbar
data = np.clip(randn(100, 100), -1, 1)
cax = plt.imshow(data, cmap=cm.coolwarm)

# Add colorbar, make sure to specify tick locations to match desired  ticklabels
cbar = plt.colorbar(cax, ticks=[-2, 0, 2])  # vertically oriented colorbar
plt.axis([-2, 2, -2, 2])
plt.title('Rosenbrock function') #add title if desired
plt.xlabel('x')
plt.ylabel('y')

plt.show()  #generate
```




Lire Comment écrire une fonction Jacobian pour optimiser.minimize en ligne: https://riptutorial.com/fr/scipy/topic/4493/comment-ecrire-une-fonction-jacobian-pour-optimiser-minimize

# Chapitre 3: Fonctions d'ajustement avec scipy.optimize curve_fit

## Introduction

L'ajustement d'une fonction décrivant l'occurrence attendue de points de données à des données réelles est souvent nécessaire dans les applications scientifiques. Un optimiseur possible pour cette tâche est curve_fit de scipy.optimize. Un exemple d'application de curve_fit est donné ci-après.

## Examples

### Adapter une fonction aux données d'un histogramme

Supposons qu'il y ait un pic de données distribuées normalement (gaussiennes) (moyenne: 3,0, écart-type: 0,3) dans un contexte en décomposition exponentielle. Cette distribution peut être équipée de curve_fit en quelques étapes:

1.) Importez les bibliothèques requises.

2.) Définissez la fonction d'ajustement à adapter aux données.

3.) Obtenir des données d'expérience ou générer des données. Dans cet exemple, des données aléatoires sont générées afin de simuler le fond et le signal.

4.) Ajoutez le signal et l'arrière-plan.

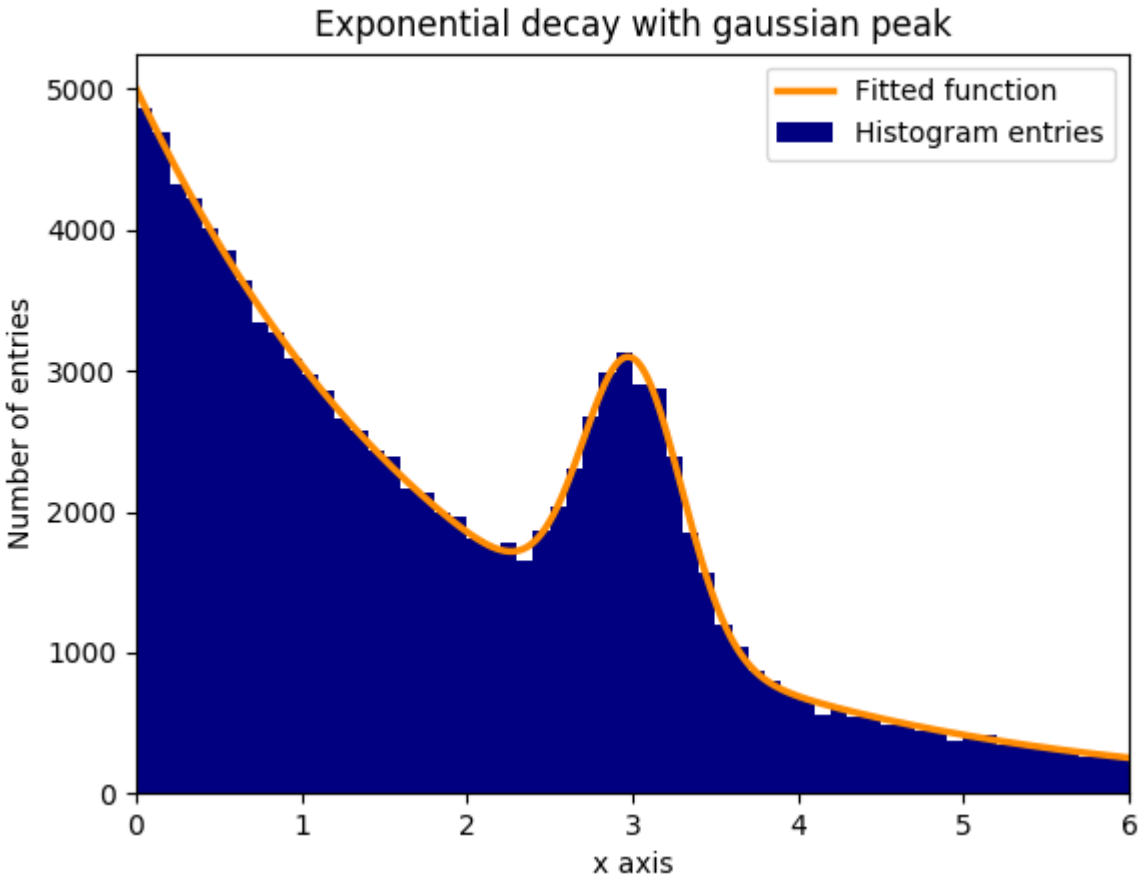
5.) Ajuster la fonction aux données avec curve_fit.

6.) (Facultatif) Tracez les résultats et les données.

Dans cet exemple, les valeurs de y observées sont les hauteurs des tranches d'histogramme, tandis que les valeurs de x observées sont les centres des `binscenters` histogramme ( `binscenters` ). Il est nécessaire de passer le nom de la fonction fit, les valeurs x et les valeurs y à `curve_fit` . De plus, un argument facultatif contenant des estimations approximatives pour les paramètres d'ajustement peut être donné avec `p0` . `curve_fit` renvoie `popt` et `pcov` , où `popt` contient les résultats d'ajustement pour les paramètres, tandis que `pcov` est la matrice de covariance, dont les éléments diagonaux représentent la variance des paramètres ajustés.

```
# 1.) Necessary imports.
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# 2.) Define fit function.
def fit_function(x, A, beta, B, mu, sigma):
```

```
    return (A * np.exp(-x/beta) + B * np.exp(-1.0 * (x - mu)**2 / (2 * sigma**2)))

# 3.) Generate exponential and gaussian data and histograms.
data = np.random.exponential(scale=2.0, size=100000)
data2 = np.random.normal(loc=3.0, scale=0.3, size=15000)
bins = np.linspace(0, 6, 61)
data_entries_1, bins_1 = np.histogram(data, bins=bins)
data_entries_2, bins_2 = np.histogram(data2, bins=bins)

# 4.) Add histograms of exponential and gaussian data.
data_entries = data_entries_1 + data_entries_2
binscenters = np.array([0.5 * (bins[i] + bins[i+1]) for i in range(len(bins)-1)])

# 5.) Fit the function to the histogram data.
popt, pcov = curve_fit(fit_function, xdata=binscenters, ydata=data_entries, p0=[20000, 2.0,
2000, 3.0, 0.3])
print(popt)

# 6.)
# Generate enough x values to make the curves look smooth.
xspace = np.linspace(0, 6, 100000)

# Plot the histogram and the fitted function.
plt.bar(binscenters, data_entries, width=bins[1] - bins[0], color='navy', label=r'Histogram
entries')
plt.plot(xspace, fit_function(xspace, *popt), color='darkorange', linewidth=2.5,
label=r'Fitted function')

# Make the plot nicer.
plt.xlim(0,6)
plt.xlabel(r'x axis')
plt.ylabel(r'Number of entries')
plt.title(r'Exponential decay with gaussian peak')
plt.legend(loc='best')
plt.show()
plt.clf()
```




Lire Fonctions d'ajustement avec scipy.optimize curve_fit en ligne:
https://riptutorial.com/fr/scipy/topic/10133/fonctions-d-ajustement-avec-scipy-optimize-curve-fit

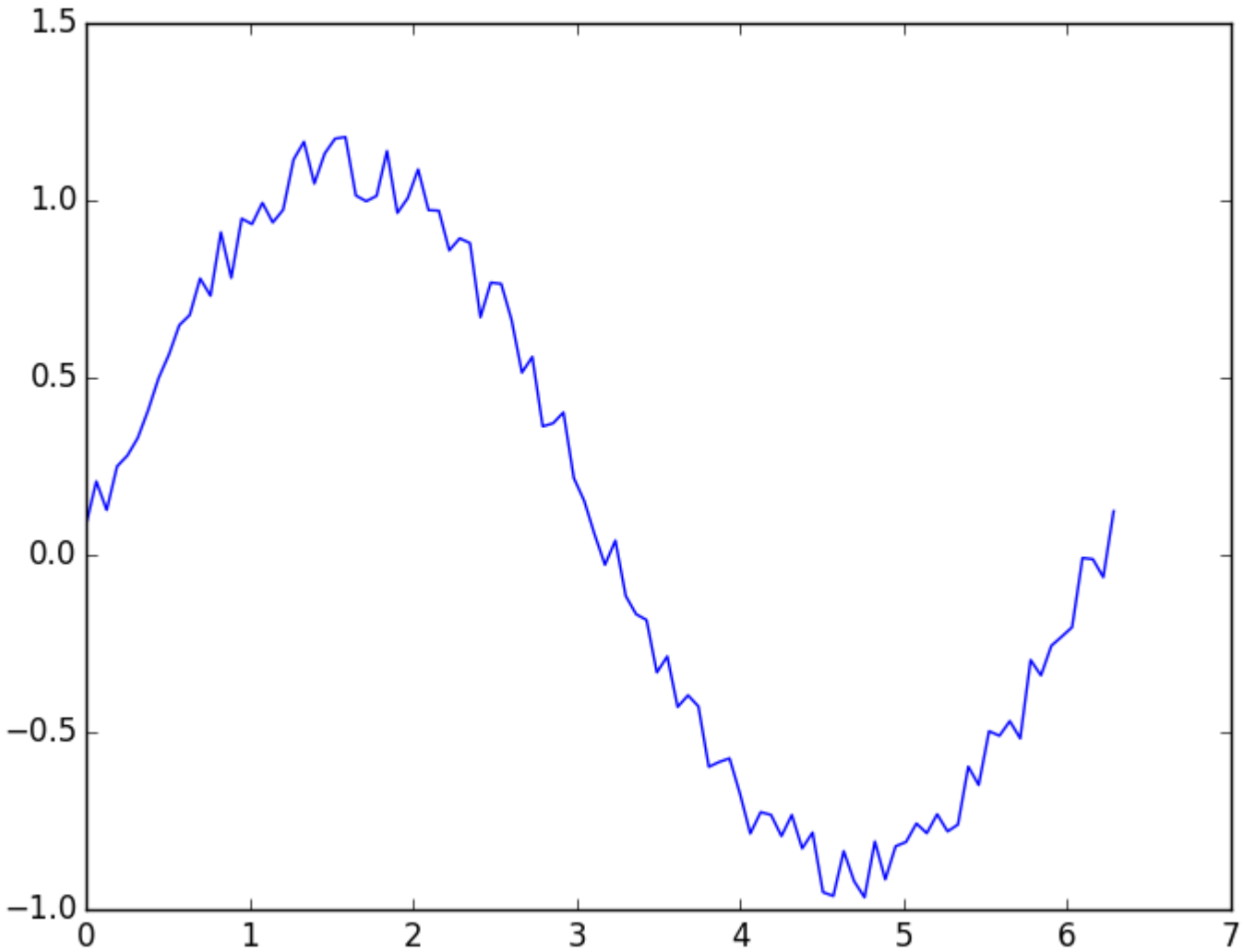
# Chapitre 4: Lisser un signal

## Examples

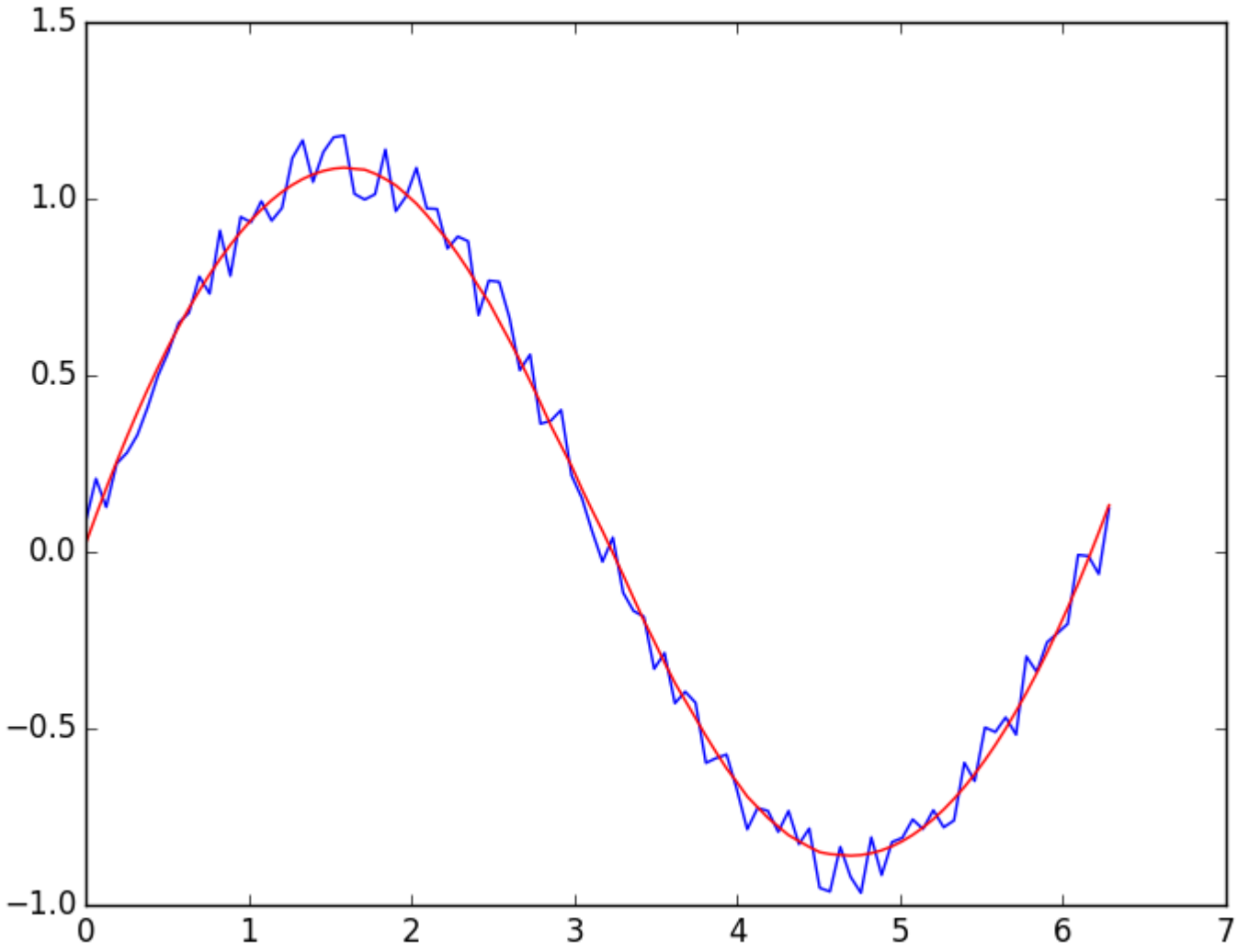
### Utilisation d'un filtre Savitzky – Golay

Vu un signal bruyant:

```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(1)

x = np.linspace(0,2*np.pi,100)
y = np.sin(x) + np.random.random(100) * 0.2

plt.plot(x,y)
plt.show()
```



on peut le lisser en utilisant un filtre Savitzky – Golay en utilisant la méthode `scipy.signal.savgol_filter()` :




```
import scipy.signal
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(1)

x = np.linspace(0,2*np.pi,100)
y = np.sin(x) + np.random.random(100) * 0.2
yhat = scipy.signal.savgol_filter(y, 51, 3) # window size 51, polynomial order 3

plt.plot(x,y)
plt.plot(x,yhat, color='red')
plt.show()
```

Lire Lisser un signal en ligne: https://riptutorial.com/fr/scipy/topic/4535/lisser-un-signal

# Chapitre 5: rv_continuous pour la distribution avec les paramètres

## Examples

### Binôme négatif sur les réels positifs

```
from scipy.stats import rv_continuous
import numpy

class Neg_exp(rv_continuous):
    def _cdf(self, x, lamda):
        return 1-numpy.exp(-lamda*x)

neg_exp = Neg_exp(name="Negative exponential", a=0)

print (neg_exp.pdf(0,.5))
print (neg_exp.pdf(5,.5))
print (neg_exp.cdf(5,.5))
print (neg_exp.stats(0.5))
print (neg_exp.rvs(0.5))
```

Il est essentiel de définir _pdf ou _cdf car scipy déduit les paramètres de l'autre fonction (que vous ne définissez pas) et l'ordre de ces paramètres dans tous les appels de fonctions que vous effectuez, à partir de votre définition. Dans ce cas, il n'y a qu'un seul paramètre de distribution, lambda. La variable représentant la valeur de la variable aléatoire apparaît en premier dans la définition de _pdf ou _cdf.

Lorsque vous définissez une seule de ces fonctions, scipy calculera l'autre numériquement. Pour une plus grande efficacité, définissez les deux. De même, définissez _stats en termes de paramètres connus pour une meilleure efficacité; sinon scipy utilise des méthodes numériques.

Notez que le support de la distribution est défini lorsque la classe est instanciée (la variable **a** est définie sur zéro et **b** est définie sur l'infini par défaut), plutôt que lorsqu'elle est sous-classée. Notez également que les paramètres de la distribution sont définis uniquement lorsque l'une des instances de classe est appelée, comme dans les cinq dernières lignes de code.

Lire rv_continuous pour la distribution avec les paramètres en ligne:
https://riptutorial.com/fr/scipy/topic/6873/rv-continuous-pour-la-distribution-avec-les-parametres

# Crédits

| S. No | Chapitres | Contributeurs |
|---|---|---|
| 1 | Commencer avec Scipy | 4444, AIB, Community, edwinksl, Rachel Gallen |
| 2 | Comment écrire une fonction Jacobian pour optimiser.minimize | Rachel Gallen, Richard Fitzhugh |
| 3 | Fonctions d'ajustement avec scipy.optimize curve_fit | ml4294 |
| 4 | Lisser un signal | Bill Bell, Franck Dernoncourt |
| 5 | rv_continuous pour la distribution avec les paramètres | Bill Bell |