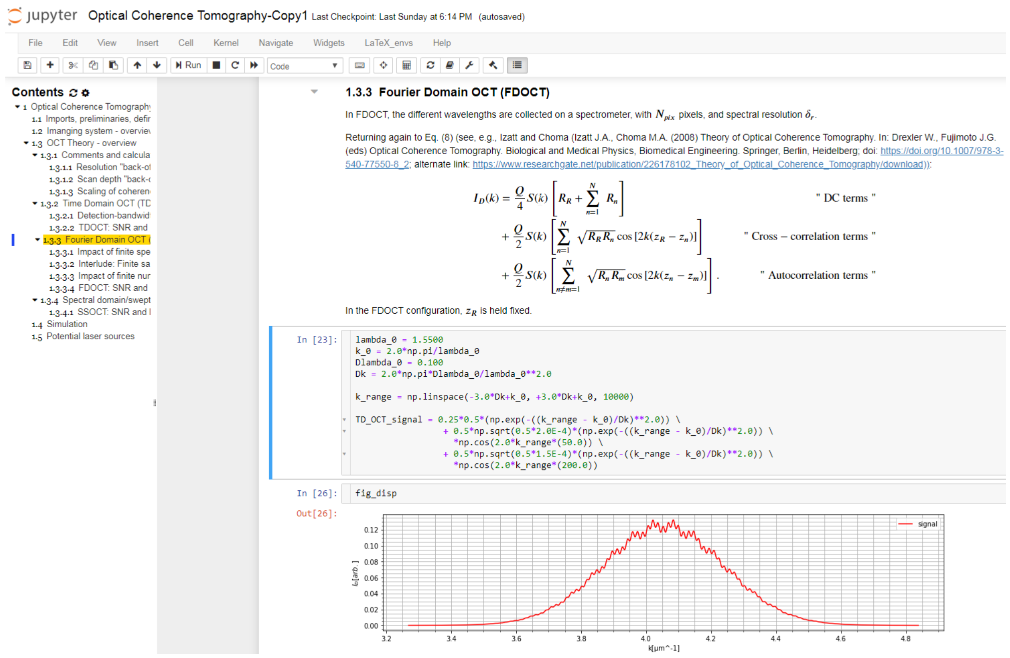
Jupyter Notebooks

Introduction

A Jupyter Notebook is a tool for interactively developing and presenting computational documents. They consist of blocks of code and markdown. The outputs of the blocks of code are a part of the block, rather than being presented at the end of the document.



Optical Coherence Tomography-Copy1 Last Checkpoint: Last Sunday at 6:14 PM (autosaved)

Contents

1.3.3 Fourier Domain OCT (FDOCT)

In FDOCT, the different wavelengths are collected on a spectrometer, with N_{pix} pixels, and spectral resolution δ_r.

Returning again to Eq. (8) (see, e.g., Izatt and Choma (Izatt J.A., Choma M.A. (2008) Theory of Optical Coherence Tomography. In: Drexler W., Fujimoto J.G. (eds) Optical Coherence Tomography. Biological and Medical Physics, Biomedical Engineering. Springer, Berlin, Heidelberg; doi: https://doi.org/10.1007/978-3-540-77550-8_2; alternate link: https://www.researchgate.net/publication/226178102_Theory_of_Optical_Coherence_Tomography/download)):

$$\begin{aligned} I_D(k) &= \frac{Q}{4} S(k) \left[R_R + \sum_{n=1}^{N} R_n \right] && \text{" DC terms "} \\ &+ \frac{Q}{2} S(k) \left[\sum_{n=1}^{N} \sqrt{R_R R_n} \cos\left[2k(z_R - z_n)\right] \right] && \text{" Cross} - \text{correlation terms "} \\ &+ \frac{Q}{2} S(k) \left[\sum_{n \neq m = 1}^{N} \sqrt{R_n R_m} \cos\left[2k(z_n - z_m)\right] \right]. && \text{" Autocorrelation terms "} \end{aligned}$$

In the FDOCT configuration, z_R is held fixed.

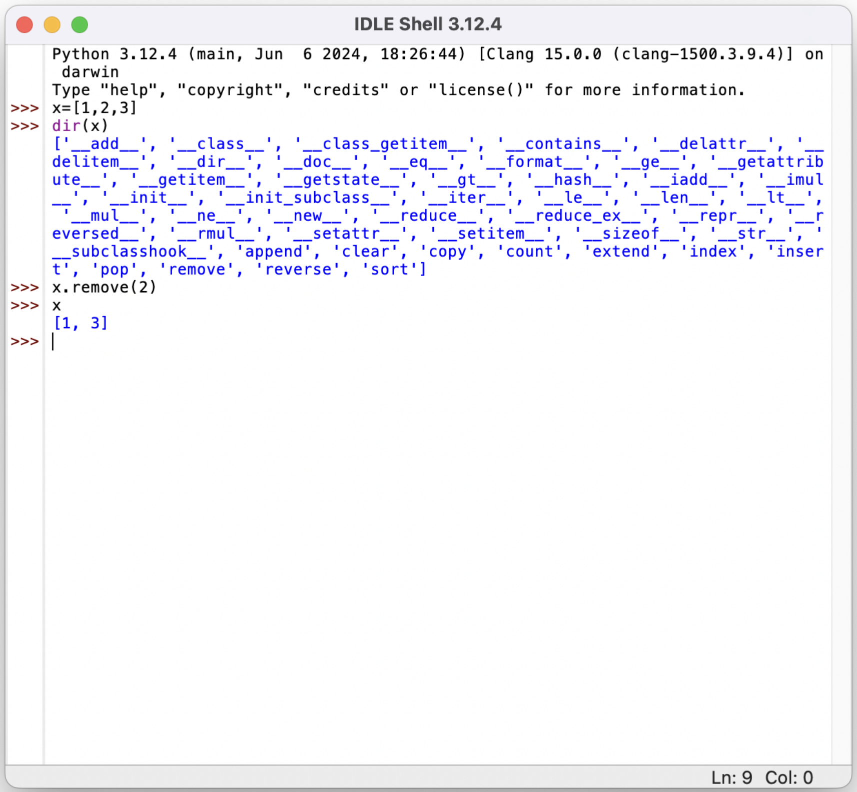
```
In [23]: lambda_0 = 1.5500
         k_0 = 2.0*np.pi/lambda_0
         Dlambda_0 = 0.100
         Dk = 2.0*np.pi*Dlambda_0/lambda_0**2.0

         k_range = np.linspace(-3.0*Dk+k_0, +3.0*Dk+k_0, 10000)

         TD_OCT_signal = 0.25*0.5*(np.exp(-((k_range - k_0)/Dk)**2.0)) \
                        + 0.5*np.sqrt(0.5*2.0E-4)*(np.exp(-((k_range - k_0)/Dk)**2.0)) \
                          *np.cos(2.0*k_range*(50.0)) \
                        + 0.5*np.sqrt(0.5*1.5E-4)*(np.exp(-((k_range - k_0)/Dk)**2.0)) \
                          *np.cos(2.0*k_range*(200.0))

In [26]: fig_disp
Out[26]:
```

Example of a Jupyter Notebook [2]

Why?

While programming, you may have jumped into a Python shell just to execute a few lines of code. For example:

1. You forgot how to remove an element from a list.
2. In a shell, you run `x=[1,2,3]`, to initialize a sample list.
3. You run `dir(x)` to list all of the attributes of `x`. You see that `remove` is a method of `x`.
4. You try `x.remove(2)`.
5. You notice that nothing is returned so you run `x` and find that the element `2` is gone. Therefore, you know that `remove` is an in-place operation on a list.



```
Python 3.12.4 (main, Jun  6 2024, 18:26:44) [Clang 15.0.0 (clang-1500.3.9.4)] on
 darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> x=[1,2,3]
>>> dir(x)
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__
delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattrib
ute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__iadd__', '__imul
__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__',
 '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__r
eversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '
__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'inser
t', 'pop', 'remove', 'reverse', 'sort']
>>> x.remove(2)
>>> x
[1, 3]
>>> |
```

The aforementioned scenario shown in the shell.

Suppose that you are exploring a data set in the same way, by quickly running code in sequential order in the shell.

1. You initialize the data set. Say it takes 10 statements.
2. You explore this data set. Perhaps you see its attributes, contents, etc.
3. After a bit of prodding, you mutate the data set and explore some more.
4. Now, you want to reinitialize the data set. Unfortunately, you would have to type out the 10 lines of code it took to prepare the data set again.

With a notebook, you can save the code for data set initialization in a single block and quickly explore the dataset in subsequent blocks (there is a keyboard shortcut for running the current code block and creating a new one right after).

Getting Started

I strongly recommend that you use an online Notebook service for learning. This will maximize reproducibility and make it easier for you to get help.

Online Notebooks

Google Colab is a free Jupyter Notebook service. Open the following link and press "New Notebook" to create a new notebook: Welcome To Colab - Colab.

Local Notebooks

I will only provide one option here, being VS Code notebook integration, and I recommend this option over Anaconda.

Install VS Code

Visual Studio Code - Code Editing. Redefined

Install Extensions

1. Install the Python extension: Python - Visual Studio Marketplace
2. Install the Jupyter extension: Jupyter - Visual Studio Marketplace

(Optional) Set Up Virtual Environment

If you would like to ensure that the packages you install for one project do not affect others, see this simple guide: Using Python Environments in Visual Studio Code.

Use Notebooks

Now, if you create or open a file with the extension `.ipynb`, VS Code should recognize it as a Jupyter Notebook.
See: Working with Jupyter Notebooks in Visual Studio Code

References

1. Jupyter Notebook: An Introduction – Real Python
2. Using the Jupyter Notebook for product prototyping - StarFish Medical