simple integrals

kacper.topolnicki@uj.edu.pl

The `integrals.py` script contains some basic examples of integrals calculated with the use of the `scipy` library. Please have a look at the official documentation (link) for more information.

Comments

The script contains many comments. Any lines that begin with the `#` symbol are ignored by python and only contain additional information for the programmer. The script file contains many instances of `#@`, `#@ref`, ... These lines are used by an external programm to create a PDF file, these lines can also be ignored by the programmer.

Running the script

To run the script simply navigate to this directory in the terminal and run:

```
<user> $ python plots.py
```

Alternatively you can make the script executable and run:

```
<user> $ ./plots.py
```

You can also run `ipython` and execute the commands one by one.

Importing the necessary libraries

First, we will import the `numpy` library (we will use this for basic mathematical functions) and the `scipy.integrate` library. [integrals.py line: 50]

```
import numpy
import scipy.integrate as integrate
```

The later will be refered to using the alias `integrate` (`import ... as ...`).

Defining functions

Next we will define a the function that will serve as the integrand [integrals.py line: 67]

```
def myFunction(x):
    return numpy.sin(x)
```

Function definitions start with the `def` keyword followed by the function name `myFunction` and a list of arguments `(x)`. The body of the function can contain a `return` statement that contains the final result of the function to be returned. In this case we are simply returning the sine (`numpy.sin`) of x.

Simple integration

We can use the `quad` method from `scipy.integrate` to instantly calculate

$$\int_0^{2\pi} sin(x)dx$$

[integrals.py line: 87]

```
print(integrate.quad(myFunction , 0 , numpy.math.pi))
```

The first artument is the function being integrated (`myFunction`), the second argument is the lower limit of the integration (`0`) and the third argument is the upper limit of the integration (π = `numpy.math.pi`). The `print` statement will write the integral and estimated error to standard output.

The `quad` function also allwows the integrated function to have additional arguments as in [integrals.py line: 100]

```
def myFunctionWithExtraParameters(x , a):
    return numpy.sin(x) + a
```

This additional argument `a` is supplied to `quad` through the optional `args` argument [integrals.py line: 105]

```
print(
    integrate.quad(
        myFunctionWithExtraParameters , 0 , numpy.math.pi , args = (1)))
```

The final example involves the normal distribution [integrals.py line: 124]

```
def normal01(x):
    return (1.0 / numpy.sqrt(2.0 * numpy.pi)) * numpy.exp(-0.5 * x**2)
```

First we will try to calculate

$$\int_{-1}^{1} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx$$

and

$$\int_{-3}^{3} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx$$

This is streightforward and results in the familiar pobabilities for 1σ and 3σ [integrals.py line: 129]

```
print(integrate.quad(normal01 , -1.0 , 1.0))
print(integrate.quad(normal01 , -3.0 , 3.0))
```

Next we will try something a little more exciting and check the normalization of our probability distribution, that is

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx$$

[integrals.py line: 142]

```
print(integrate.quad(normal01 , -numpy.inf , numpy.inf))
```

The `scipy.integrate` library can handle the concept of infinity (represented using `numpy.inf` from the `numpy` library).