Curve fitting

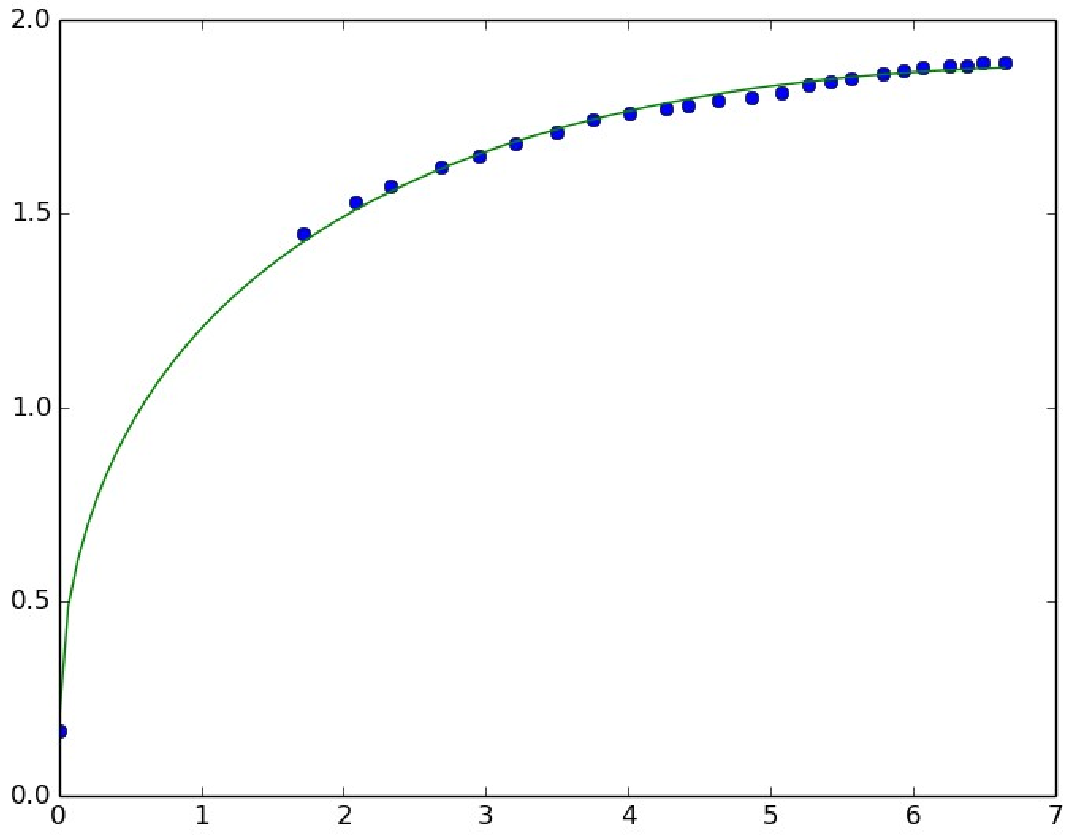
The problem:

You have a nonlinear sensor that gives you some digital or analog values that you want to convert into a result that makes sense for what you want to measure. In our case it was the voltage of a Wheatstone bridge that should be converted to a wind velocity.

So you do some measuring and get a table of values representing input and output of our converting function.

There are different ways to solve this problem, but with Pythons numpy module, it is very easily done.

Example:



The blue points are measured values, the curve is fit in this example by a function of the form $f(x) = a + b \cdot x + c \cdot \sqrt{x}$

The curve fitting function can be freely defined e.g. as a polynome or any other function.

How is it done?

Here is the code for another, simpler example:

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

# Measured values:
x = np.array([1, 2, 3, 4, 5])
y = np.array([1, 4.3, 9.2, 17, 24.8])

# define fit function:
def fit_func(x, a, b, c):
    return a + b*x + c*x**2

# optimize parameters a, b, c for best fit function:
params = curve_fit(fit_func, x, y)

# get coefficients:
[a, b,c] = params[0]
print "Fit function:\n"
print a, " + ", b, " *x + ",c, " * x**2\n"


# plot points
plt.plot(x,y,"o")

# plot fit_function
nb_x=len(x)                           # number of x values
x=np.linspace(x[0],x[nb_x-1], 100)    # linspace from first to last x, 100 values
yf=fit_func(x, a, b, c)
plt.plot(x, yf)
plt.show()
```

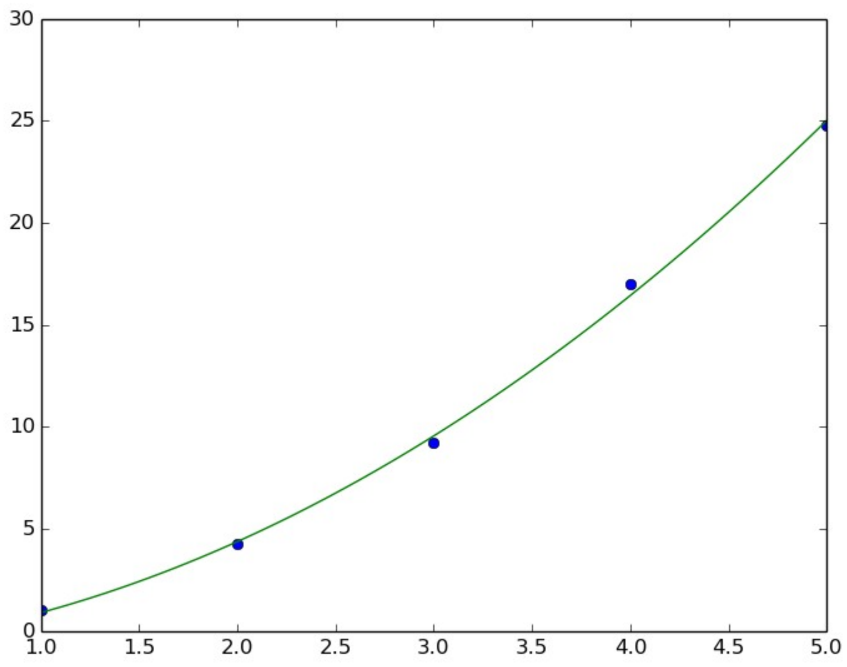
In this example the x, y values are very close to a pure quadratic function.
So, the fit function is defined as a polynomial of second degree.

The optimization of the coefficients a, b, c is done by
`params = curve_fit(fit_func, x, y)`

When this is done, the values of the parameters a, b, c are stored in **`params[0]`**

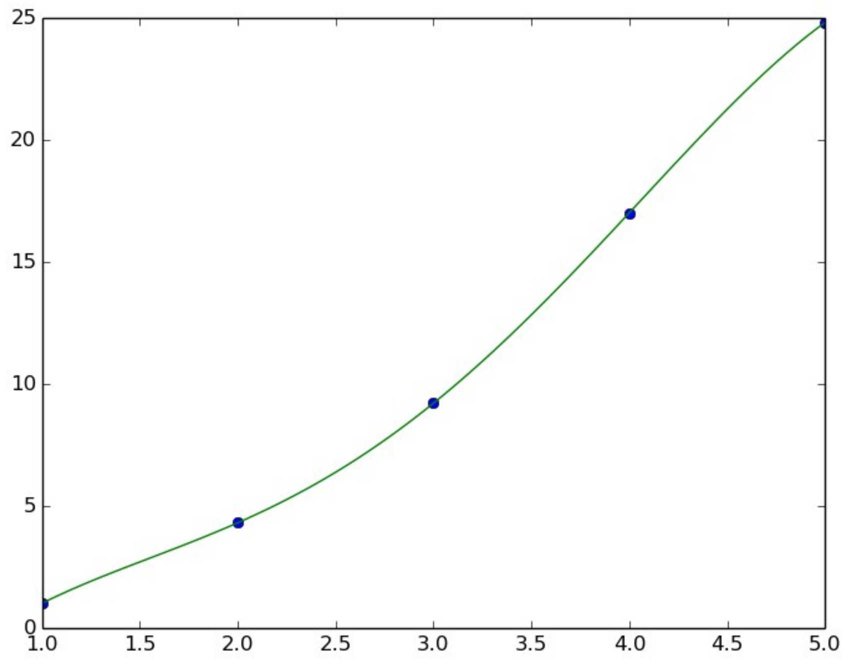
The second part of the script is only for plotting the points and the fir function.

Results:
```
-0.88  +  0.93  *x +  0.85  * x**2
```



It is good to have already an idea about the structure of the fitting function.
Using higher polynomials can lead to undesired oscillations in the curve.

The next diagram shows the same points fitted by a polynomial of 4th degree:



The curve passes well (even better) through all points, but maybe the simpler fitting function would be to be preferred, if we knew that the behaviour is closer to the second degree and there are some measurement errors.
So it is important to have some knowledge about the kind of function that would fit best.