
Python DeCal
Week 6

Announcements

- 4th Hw was due just now!
- Office Hour
  - Sorry for forgetting the office hour T_T
- Attendance!
  - https://tinyurl.com/go-bears-NL

Announcements

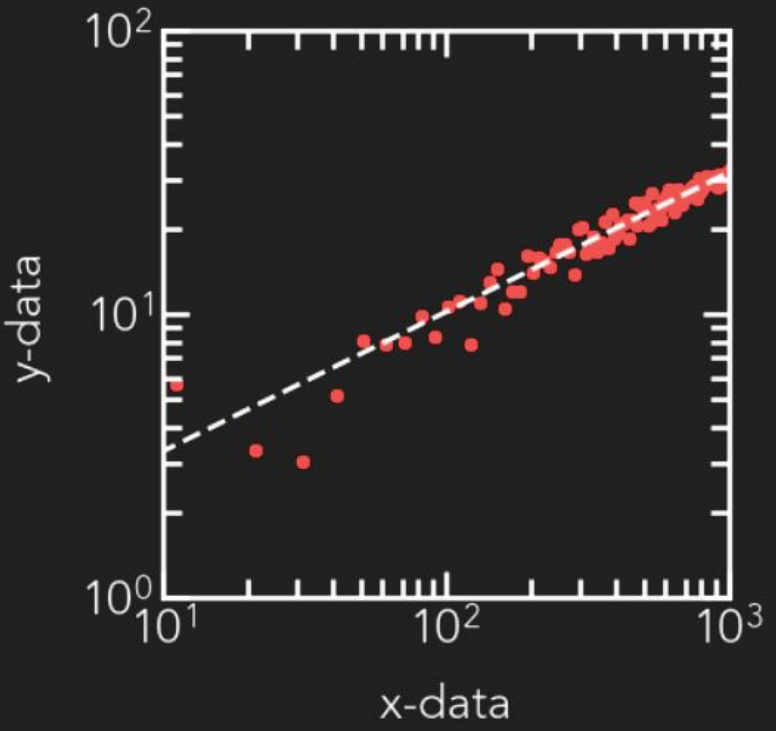
- Final Projects
  - Please complete the bcourse “quiz” so we can pair y’all up with a lab partner
  - Final Project Proposal due: **Oct 23**
- Special Topics
  - Let us know what you want to learn~
  - Sample topics: Machine learning, Many body simulation, Observational astronomy

Recap

- What are some examples of parameters we can put in `plt.plot()` function?
- What types of plot other than `plt.plot()` and `plt.scatter()` can you plot with matplotlib?

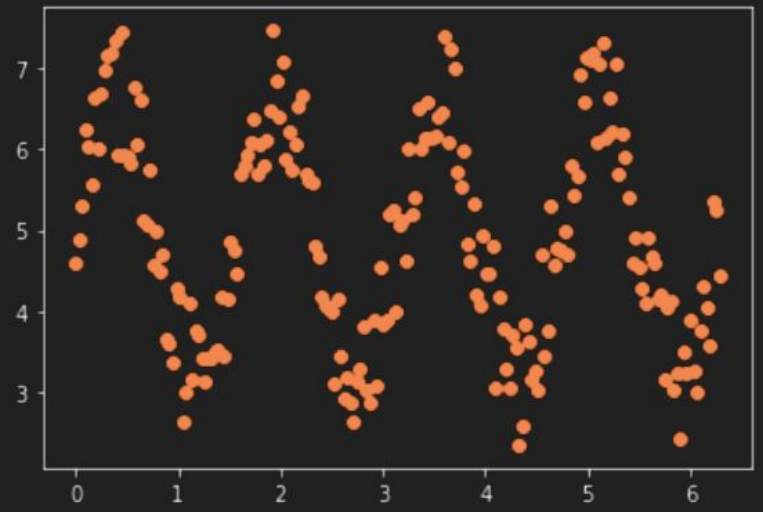
How do we better analyze data after graphing them?

- We can fit the dots/data with a curve!
- This will let us know the general trend of the data as well as give statistical insights



What do we need for fitting?

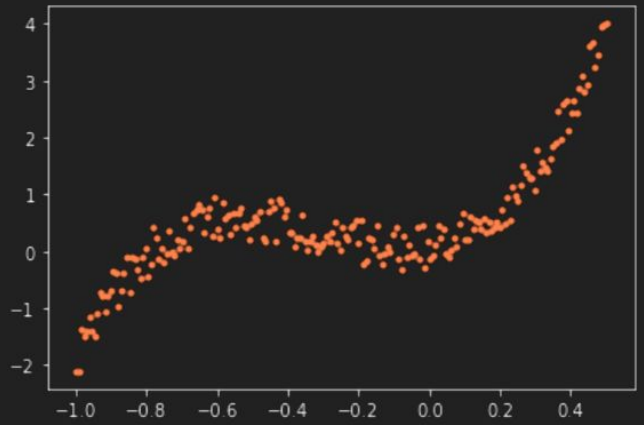
DATA



MODEL

$$y = f(x)$$

Again, start with the easiest... POLYNOMIALS!!!



$$y = a_0 + a_1 x + a_2 x^2 + \ldots + a_P x^P = \sum_{i=0}^{P} a_i x^i$$

Degree of the polynomial
(the highest power P)

```
np.polyfit(x, y, deg)
```

It outputs an array of coefficients

```
[aP, … , a1, a0]
```

What is a good fit?

- Minimise chi-squared

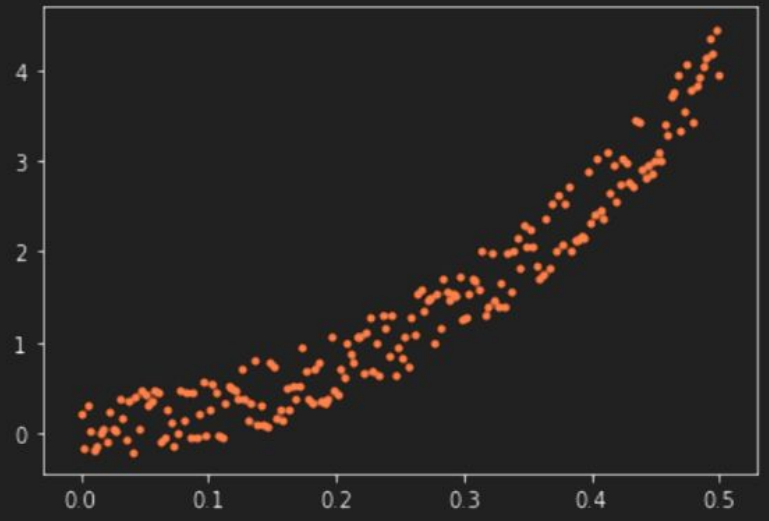
Fitted model

$$\chi^2 = \sum_i \left(\frac{y_i - f(x_i)}{\sigma_i} \right)^2$$

uncertainty

- Be aware of overfitting
  - Number of data points should be much much much much much much larger than the number of fitted parameters

What function could be the model?



Now... More complicated functions...

```
from scipy.optimize import curve_fit
```

- Importing the function this way will allow you to directly use the curve_fit function without typing everything else.
- This command basically picks out the function from scipy.optimize package.

The `curve_fit` function

```
fit_params = curve_fit(model, xdata, ydata, \
                       p0=[...], \
                       sigma=[...], \
                       bounds=([...], [...]))
```

Initial guess, [...] is an array

Uncertainty on ydata, [...] is an array of length len(ydata)

Fixes range for your fitted parameter

A note on defining the model function:

```
def model(x, a1, a2, a3):
    return a1*np.sin(x)**a2 + a3
```

Fitted parameters starting from the 2nd position

Fitted parameters can be obtained by calling `fit_params[0]`

Finding the roots...

$$\sqrt{1-x^2} = \sin(x)$$

- There are equations we simply cannot solve analytically by hand

```
from scipy.optimize import root

sol = root(func, x0)
```

```
def func(x):
    return np.sqrt(1-x**2)-np.sin(x)
```

The equation you want to find roots of

Initial guess (can be found by plotting)

- Solution can be obtained by calling **sol.x**

Finding the roots...

```
def func(x):
    return np.sqrt(1-x**2)-np.sin(x)
```

- We can also use the function from scipy.optimize:

```
from scipy.optimize import fsolve

fsolve(func, x0, args=())
```

- What if you are given this function for the first time? Go to breakout rooms and discuss with others and come back with the explanation for the highlighted variables!

Finding the roots...

```
fsolve(func, x0, args=())
```

- **func**: a callable function that takes in at least one arguments and returns output of the same length.
- **x0**: The starting estimate of the roots of func(x)=0
- **args**: any other arguments that goes to func that you have defined.
- Outputs an ndarray that contains the solution.

Attendance: https://tinyurl.com/triangle-orbit

Friday

Recap

- What are the differences between **`curve_fit`** and **`np.polyfit`**?
- Challenge: Why did the **`curve_fit`** function not fit for Yilun?

Differentiation...

- Again, we use some packages

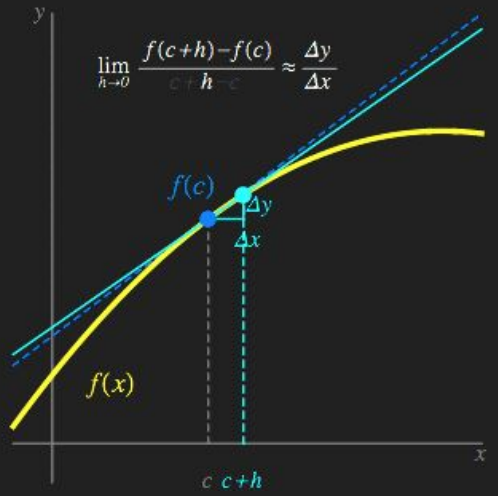
```
from scipy.misc import derivative

derivative(func, x0, dx, n)
```

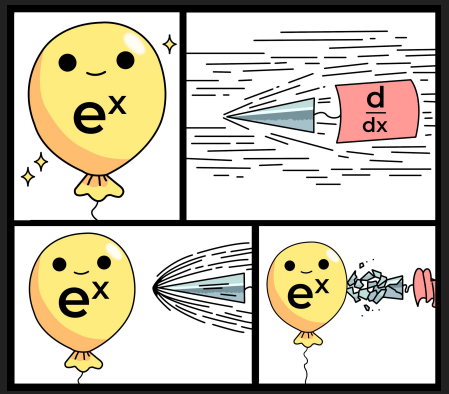
You have to define the function that you want to integrate

The point where you want to evaluate the derivative

spacing

Order of differentiation





Two ways to define the function

- The normal way:

```
def func(x):
    return  x**2 + x
```

- Lambda

```
derivative(lambda x: x**2+x, x0=17, dx=1e-6, n=1)
```

Writing a short function of variable x

Numerical Integration

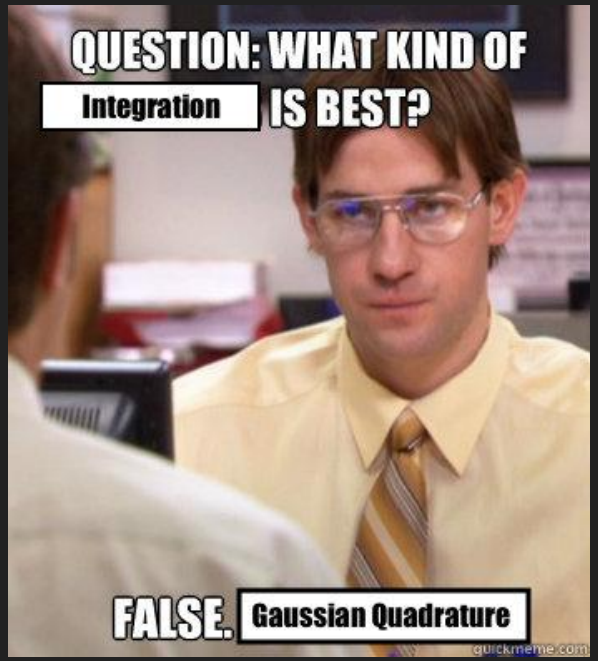
Riemann Sums (Remember those???)

-Centered, Left, and Right Riemann Sums

-Trapezoidal Rule

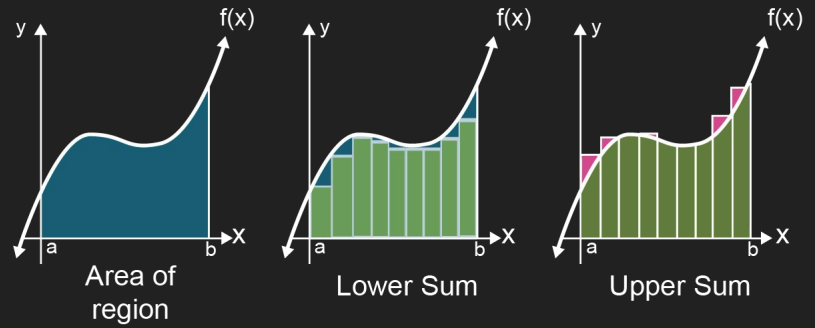
-Simpson's Rule (what's that?)

-Gaussian Quadrature (what's that?)



While-loop integration

Demo

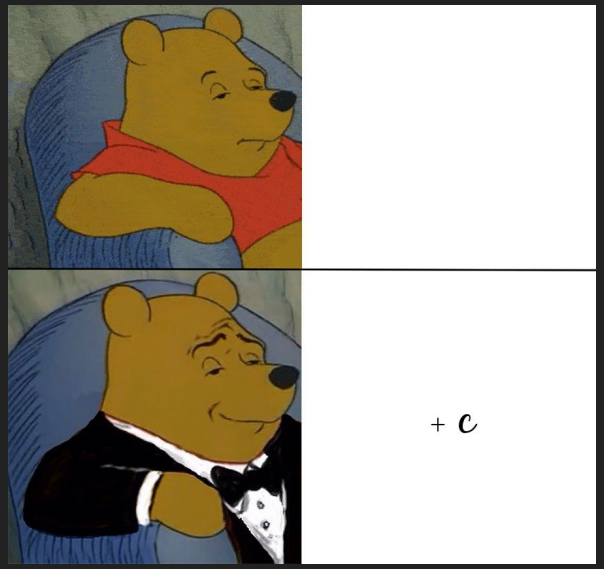


Calcworkshop.com

$$\int_a^b f(x)\,dx = \lim_{n\to\infty} \sum_{i=1}^{n} f(x_i)\Delta x \qquad \Delta x = \frac{b-a}{n} \qquad \text{and} \qquad x_i = a + i\Delta x$$

Numerical Integration

- Extremely important in physics as well as astronomy and all natural science tbh…



- This time we don't import a function,
- We import an entire package

```
import scipy.integrate as integrate

from scipy.integrate import ...
```

Slightly different…
But either way

Options

`Scipy.integrate` has tons of different options

Integrating functions, given function object

quad(func, a, b[, args, full_output, ...])	Compute a definite integral.
quad_vec(f, a, b[, epsabs, epsrel, norm, ...])	Adaptive integration of a vector-valued function.
dblquad(func, a, b, gfun, hfun[, args, ...])	Compute a double integral.
tplquad(func, a, b, gfun, hfun, qfun, rfun)	Compute a triple (definite) integral.
nquad(func, ranges[, args, opts, full_output])	Integration over multiple variables.
fixed_quad(func, a, b[, args, n])	Compute a definite integral using fixed-order Gaussian quadrature.
quadrature(func, a, b[, args, tol, rtol, ...])	Compute a definite integral using fixed-tolerance Gaussian quadrature.
romberg(function, a, b[, args, tol, rtol, ...])	Romberg integration of a callable function or method.
quad_explain([output])	Print extra information about integrate.quad() parameters and returns.
newton_cotes(rn[, equal])	Return weights and error coefficient for Newton-Cotes integration.
IntegrationWarning	Warning on issues during integration.
AccuracyWarning	

Integrating functions, given fixed samples

trapz(y[, x, dx, axis])	Integrate along the given axis using the composite trapezoidal rule.
cumtrapz(y[, x, dx, axis, initial])	Cumulatively integrate y(x) using the composite trapezoidal rule.
simps(y[, x, dx, axis, even])	Integrate y(x) using samples along the given axis and the composite Simpson's rule.
romb(y[, dx, axis, show])	Romberg integration using samples of a function.

Single-variable integral....

$$\int_a^b f(x)\,dx$$

```
from scipy.integrate import quad
```

```
integral = quad(func, a, b)
```

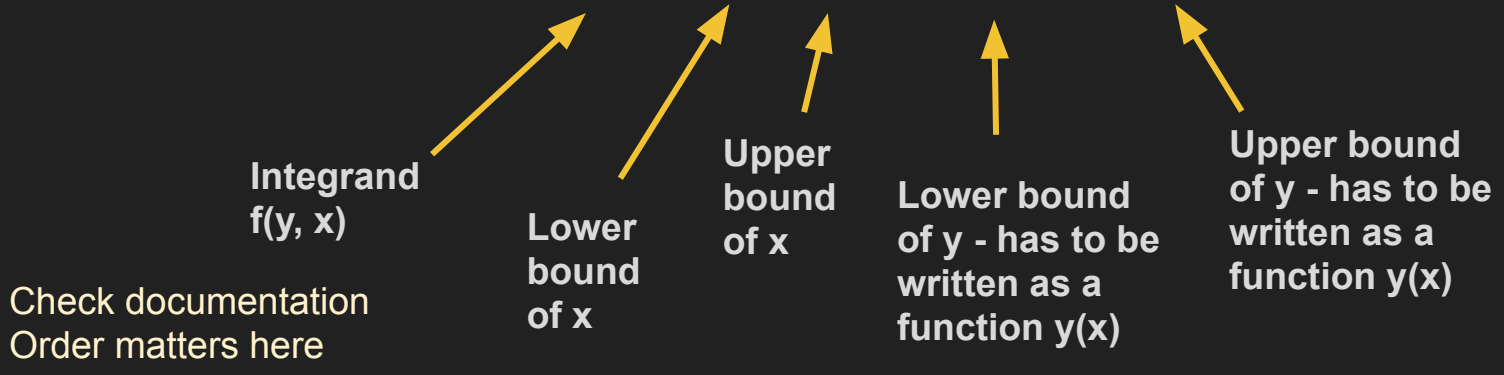
Result can be extracted by calling `integral[0]`
Error can be extracted by calling `integral[1]`

Double integral....

$$\int_{x_1}^{x_2} \int_{y_1}^{y_2} f(x,y)dydx$$

```
from scipy.integrate import dblquad
```

```
integral = dblquad(func, a, b, gfun, hfun)
```



Double integral example

$$\int_3^6 \int_1^{x^2} xy \; dydx$$

```
def func(y, x):
    return x*y
```

```
I = dblquad(func, 3, 6, lambda x:1, lambda x: x**2)
```

Similarly, you can also do a triple integral using `tplquad`