CHE502 Chemical and Statistical Thermodynamics
Topic 4
Computational aspects, fitting, minimization, random numbers

The book of nature is written in mathematics.
Without it, it is impossible to understand anything and one risks getting lost in an obscure labyrinth

Galileo Galilei (1564-1642)

URL: http://theo.ism.u-bordeaux.fr/PBopp/vistec.2020.2021

As we have seen, it is quite easy to write (e,g,) a Hamiltonian $\hat{H}$ for an S.eq. eg.

$$\hat{H} = \hat{T} + \hat{V} = \sum \frac{p^2}{2m} + \sum \frac{q_i q_j}{r_{ij}} + ...$$

It is, however, generally (ie. except in some very simple cases, see table on page 7 of topic 2), **impossible** to solve such equation(s), i.e. to find mathematical expressions
E_i = something and ψ_i=something.
(The same is true for the equivalent Newton's equations)
This is not because we don't know how to do the math, it can be
demonstrated rigorously that it is not possible.

One thus has to use clever approximations ($\rightarrow$ quantum chemistry, $\rightarrow$ simulations (CHI501))

To do these approximations for "chemically interesting" systems
(something a little more complex than the H atom, or the H_2^+ ion, or the H_2 molecule,
not to mention many molecules), one needs, among other things,
- a lot of very smart programming
- a lot of computer time (hours, days)
- careful and critical scrutiny of the computed results

Fortunately, there are many good and free codes (programs) available for many types of calculations which are of interest in chemistry and chemical engineering
(*There are also many expensive, but often not so good, commercial "software suites"*)

In both cases, one must always be critical of the results
Free software is often 'open source', ie. one can look inside and knows exactly what the program is doing. One can also modify it if necessary
For commercial codes one can usually not verify what they do 'inside'
Such programs are called 'black boxes'

GIGO "Garbage In - Garage Out". This means any program will give you a nonsense output (result) if you supply a nonsense input (data)

It is thus always difficult to assess whether a program does what you think it does, or should do, and whether it does it consistently and reliably

This is why you always need a scientific background to check on your computed results

As simple example for this, we shall look at two applications:

- Fitting
- Minimization
- Monte Carlo Methods

Fitting

We have a number N of (x_i, y_i) points and want to find the coefficients of a function $y = f(x)$ that is the "best possible" (?) representation of the points.

– We invent a function $f(x)$ that we think should be suitable, eg a linear one $f(x) = a + b \cdot x$ with the two parameters a and b to be determined.
– We must decide: what is "best possible"
Choice 1: the (vertical) distances between y_i and $f(x_i)$ should be minimal,
we try to find the minimum of

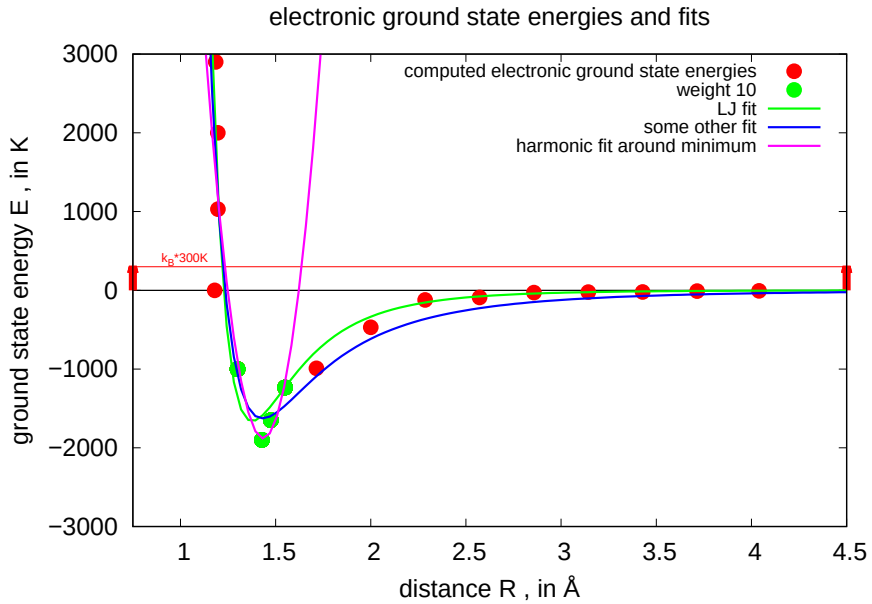
$$\sum_i^N \mid y_i - f(x) \mid \quad \text{or} \quad \sum_i^N (y_i - f(x))^2$$

or of the relative error

$$\sum_i^N \mid \frac{(y_i - f(x))}{y_i} \mid$$

or any other criterion we may deem useful, taking eg into account that there may be different Δys and Δxs in our data points

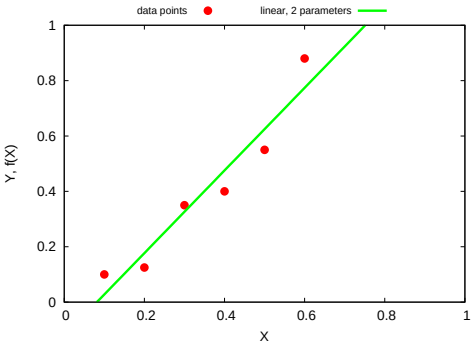
One trick is to give data points deemed more important, or with smaller uncertainties, a higher weight.
The figure shows how the fits shown in Topic 3, page 9 change (a bit) when the points around the minimum are given a higher weight.

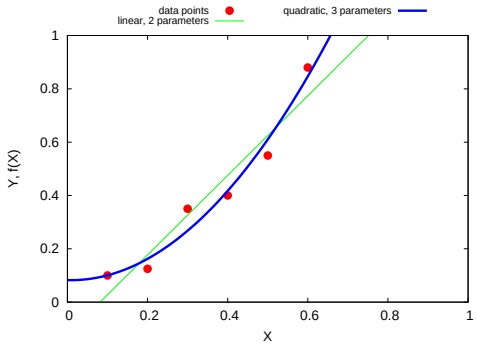


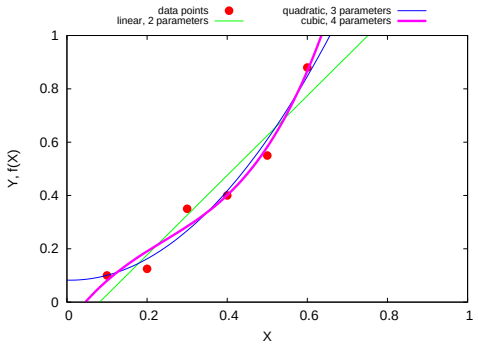
"Fitting" is a particular case of a more general mathematical problem:
finding extrema (minima, maxima) of complicated functions
We will come to this later

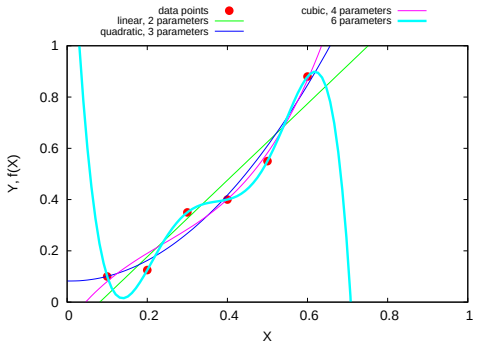
Of course, when we have N points, we can always find a function that will fit the N points exactly (eg a polynomial of order N). Such fits are usually meaningless and useless

The sense and nonsense of fitting

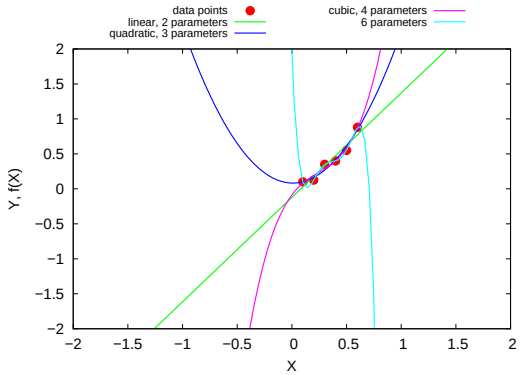








same fits, different ranges



The mathematically 'best' fit is not necessarily the physically 'best' fit (overfitting)
One must carefully select the fitting function $f(x)$, best based on a physical model

These plots and fits were made with the free software `gnuplot`*, which is highly recommended*

How do fit programs work

Linear fit vs. **non-linear** fit

Linear: the fitting coefficients to be determined (eg $c_1, c_2, c_3, ...$) appear only linearly in $f(x)$
(or $f(x, y, z, ...)$, eg $f(x) = c_1 \cdot x + c_2 \cdot x^2$
This works almost always (except for numerical problems), there is an analytical solution
see https://en.wikipedia.org/wiki/Linear_regression for details

Non-linear: the fitting coefficient appear non-linearly, eg $f(x) = \cos(c_1 \cdot x) * \exp(-c_2 \cdot x)$
see https://en.wikipedia.org/wiki/Nonlinear_regression
This can be tricky, even very tricky, there are many methods (*may be we can look at a few*)
convergence may be difficult to achieve, one finds many local minima, but
not the global one, etc etc

→ gradient methods
→ Monte Carlo methods
→ neural networks
This is an active field of research

Famous historic examples of minimization problems

– The traveling salesman problem
  A salesman has to visit N customers in different places,
  How can he minimize his travels
  https://en.wikipedia.org/wiki/Travelling_salesman_problem

related:
– The Königsberg Bridge Problem (Immanuel Kant (1724-1804))
  Kant wanted to find out whether he could make a walk starting from his home and returning to his home, crossing each bridge only once
  https://en.wikipedia.org/wiki/Seven_Bridges_of_Königsberg

The most famous story about reproducing experiments accurately and correct descriptions

– The earth is the center of the universe. Since Ptolemy $\approx 100 - 170$ there were mathematical expression (cycles,epicycles ...), which, based on this view, described and predicted the positions of the (then known) planets reasonably well

– Nicolaus Copernicus (1474-1543) said: The sun is in the center.
But this model did not always give the planet positions very accurately

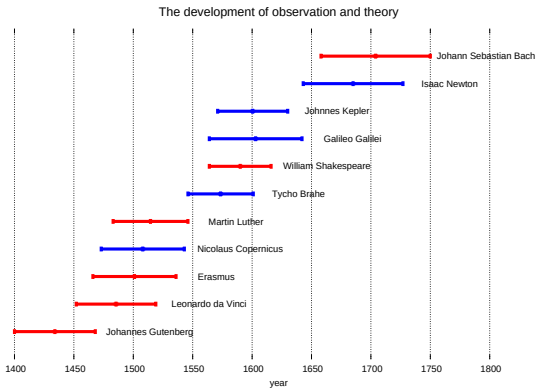
– Tycho Brahe (1546-1601) re-measured the planet motions extremely carefully. He concluded that Copernicus was wrong and made a new, very sophisticated model (some planets go around the sun, others go around the earth)

– Johannes Kepler (1571-1630) said: The planetary orbit are not circles, but ellipses.
This fitted Tycho's data well.

– Galileo Galilei (1564-1642) observed Jupiter's moons going around their planet

– Isaac Newton (1643-1727) found the law that underlies all these observations
(and the apple falling on his head)

Note:
Ptolemy was Greek (or Egyptian), Copernicus was Polish (or German), Tycho was Danish, Kepler was Austrian, Galileo was Italian, Newton was English, science always was international



Random numbers

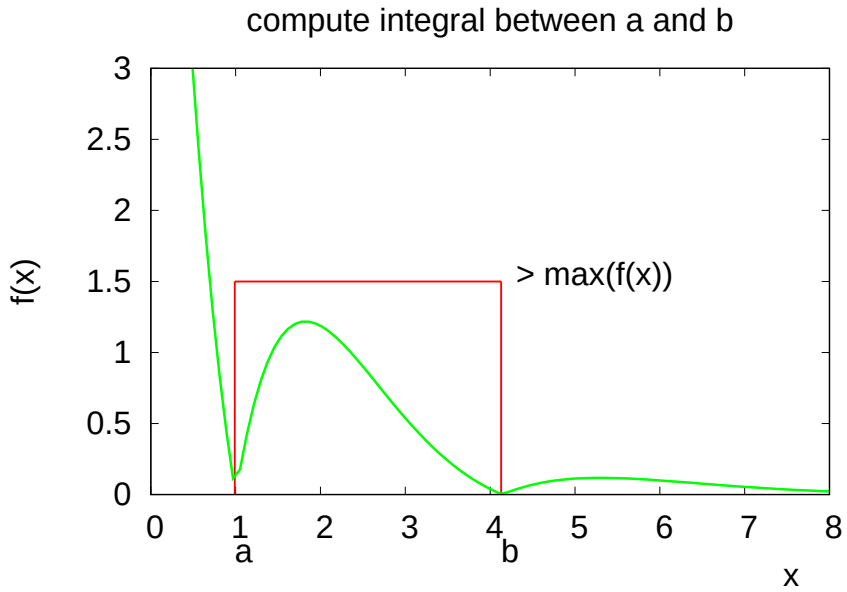
see https://en.wikipedia.org/wiki/Random_number

(Pseudo-)random numbers are used in many ways in numerical mathematics
We will look at a few applications:
1) Calculating integrals
2) Fitting
3) Monte Carlo (MC) Simulations (in CHE501)

1) The best known example is:
   You are given a pseudo random number generator that generates
   real number equidistributed between 0 and 1
   Determine the value of π

http://www.stealthcopter.com/blog/2009/09/python-calculating-pi-using-random-numbers/
https://theabbie.github.io/blog/estimate-pi-using-random-numbers.html
and many other sites

For a general integral $\int_a^b f(x)\ \mathrm{d}x$ you embed $f(x)$ in a surface $(b-a)\cdot\max(f(x))$, scale and shift your random numbers (to be between a and b for x and between 0 and max$(f(x))$ for y) and count the random points below $f(x)$ $\Rightarrow$ next page



The number of random points below the curve between a and b, divided by the total number of points, is proportional to the surface under the curve (ie its integral) and the surface of the red square

2) Fitting

Finding a minimum (or a maximum) in many-dimensional space is very difficult

Example: We have an energy E that depends on the positions of many (10, 100, 1000) particles (molecules):

$$E(x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3, x_4, y_4, z_4, x_5, y_5, z_5, ...)$$

Question: for which positions is the energy minimal?

One way to do it: random moves (Monte Carlo)
(i) Select (very carefully!) a starting point $(x_1^0, y_1^0, z_1^0, x_2^0, y_2^0, z_2^0, x_3^0, y_3^0, z_3^0, ...)$
(ii) Change one variable (coordinate) randomly, not too much and not too little
(iii) Recompute the energy with the changed coordinate
(iv) If the new energy is lower, keep the new (changed) coordinate,
if not, keep the old coordinate
(v) go back to (ii)

This is a 'trial and error' method
it will probably find a minimum,but not necessarily the global (lowest) one

3) Monte Carlo (MC) simulations

Many methods using random numbers are called 'Monte Carlo', like eg the minimum search method we just discussed

This is not to be confused with the Monte Carlo simulations in statistical mechanics that we will discuss in che501