# Numerical methods notes

Jacopo Tissino, Giorgio Mentasti

November 28, 2019

The course is held by Michela Mapelli: a team leader in GW astronomy.

The first lessons are about basic Linux and Python, I will not take notes now, I will start later on.

## Thu Nov 07 2019

## 1 Sorting

Useful for astrophysics since we often deal with large files.

**Bubble sort** We loop through the file, and swap each pair if it is not ordered.

It is $O(n^2)$ in general.

**Selection sort** We look for the minimum and put it at the beginning, then scan the remaining array.

It is $O(n^2)$ in general.

**Quicksort**

1. We pick an element, the *pivot*;
2. we reorder the array so that all elements less than the pivot come before it;
3. we do this recursively to the subarrays to the left and right of the pivot.

It is $O(n^2)$ in the worst case, $O(n \log n)$ usually.

**Merge sort** We divide the array into small subarrays, and merge them to produce larger subarrays.

It is $O(n^2)$ in the worst case, $O(n \log n)$ usually.

# Fri Nov 08 2019

One can time a bash command using

```
time [Command]
```

# 2 Linear systems

We want to solve a linear system, in the form $A\vec{x} = \vec{b}$, with unknown $\vec{x}$. How do we solve this numerically? We can transform our system to an equivalent one, by

1. exchanging two rows;
2. multiplying an equation by a nonzero constant;
3. adding an equation to another.

These allow us to do Gaussian elimination, and LU decomposition. These are *direct methods*.

Another class is that of *indirect methods*: we start with an *ansatz* and refine it. These are easier to implement, more generally applicable, more efficient if the matrix is sparse. They, however, do not always converge.

An example is the Gauss-Seidel method.

# Thu Nov 14 2019

## 2.1 Gauss-Seidel

We can rewrite $\sum_j A_{ij}x_j = b_i$ as

$$x_i = \frac{1}{A_{ii}}\left(b_i - \sum_{j\neq i} A_{ij}x_j\right), \tag{1}$$

and the algorithm works by starting with and *ansatz*, updating it with this formula, and iterating. The update can be written more generally as

$$x_i^{n+1} = \frac{\omega}{A_{ii}}\left(b_i - \sum_{j\neq i} A_{ij}x_j^n\right) + (1-\omega)x_i^n, \tag{2}$$

with the *relaxation parameter* $\omega$. Do note that $n$ is not an exponent but an iteration number.

A good choice for $\omega$ after the 5th iteration:

$$\omega_{\text{opt}} = \frac{2}{1+\sqrt{1-(\Delta x^{k+p}/\Delta x^k)^{1/p}}}\,, \tag{3}$$

# Fri Nov 15 2019

Implementation of Gauss-Seidel.

# Thu Nov 28 2019

We want to generate random numbers in order to draw samples from known distributions. We can only really generate *pseudo*-random numbers: for example, we vary the integer$x$ in the formula

$$x' = (ax + c) \mod m\,, \tag{4}$$

with constant integer $a$, $c$ and $m$.

Importantly, the values of the constants and the starting value of $x$ constitute a *seed* which can be used to reproduce our results.

If we want to produce numbers distributed according to an arbitrary pdf, we first produce uniformly distributed numbers, and then use the laws of probability. Say we have a desired pdf, $p(x)$, and numbers uniformly distributed from 0 to 1, then

$$\int_{-\infty}^{y(x)} p(\widetilde{y})\,\mathrm{d}\widetilde{y} = \int_0^x \mathrm{d}\widetilde{x} = x\,, \tag{5}$$

so if we are able to solve this and make $y(x)$ explicit as a function of $x$ we are done! The final expression is given in terms of the cumulative pdf:

$$y = P^{-1}(x)\,. \tag{6}$$

We want to generate numbers $m$ between 0.1 and 150 distributed according to $p(m) = m^{-\alpha}$.

The cdf is

$$\frac{1}{N}\int_{0.1}^{m} m^{-\alpha}\,\mathrm{d}m = \frac{1}{N}\left(-\frac{0.1^{-\alpha}}{1-\alpha} + \frac{1}{1-\alpha}m^{1-\alpha}\right), \tag{7}$$

where the normalization is

$$N = \int_{0.1}^{150} p(m)\,\mathrm{d}m = \frac{1}{1-\alpha}\left(150^{1-\alpha} - 0.1^{1-\alpha}\right), \tag{8}$$

therefore we have

$$Nx = \left( -\frac{0.1^{-\alpha}}{1-\alpha} + \frac{1}{1-\alpha} m^{1-\alpha} \right), \tag{9}$$

so

$$m = \left( \left( 0.1^{1-\alpha} + (1-\alpha)Nx \right) \right)^{1/(1-\alpha)}, \tag{10}$$