

# Calculation methods applied to chemistry

## Synopsis (english)

### Mathematical prerequisites

### Programming bases and tools

- Python programming language
  - [LearnPython.org](#) interactive tutorial with code execution
  - [DataCamp free course "Intro to Python for Data Science"](#)
  - [Python 3 Tutorial](#), interactive, with [code use in web browser](#)
  - MOOCs (massive open online courses) :
    - [An Introduction to Interactive Programming in Python \(Beginners\)](#) (Coursera)
    - [Introduction to Computer Science and Programming Using Python](#) (edX)
-  [Anaconda Python distribution](#) (64 bits, with Python 3.x) :  
<https://www.anaconda.com/download/>
  - Includes these tools :
    - Jupyter notebook (interactive web-based environment)
    - qtconsole (high level Python console with graphics & colors)
    - spyder (powerful Python IDE)
  - includes lot of Python libraries : matplotlib, numpy, scipy, pandas,...
  - package management system (conda), virtual environments
  - Anaconda Navigator includes extensive documentation (on anaconda website and dedicated websites)
- GNU/Linux OS (preferred)
- Jupyter introductions, tutorials, ...
  - [Jupyter Notebook Tutorial](#), par Den Kasyanov (Medium)
  - [Jupyter official documentation](#)
  - [Jupyter DataCamp Cheat Sheet](#)

The Microsoft Azure Notebooks environment can be used to execute sample codes, using a professional, personal or student login/account (i.e. student login from UMONS). The following public sample notebooks are then available to test Jupyter text and coding features :

- <https://notebooks.azure.com/linusable/libraries/samples-public>
  - e.g. [https://notebooks.azure.com/linusable/libraries/samples-public/html/notebooks/second\\_degree\\_polynomial\\_and\\_roots-01.ipynb](https://notebooks.azure.com/linusable/libraries/samples-public/html/notebooks/second_degree_polynomial_and_roots-01.ipynb) (Second degree polynomial and roots)

- Python scientific libraries (official websites including tutorials and documentation)
- cf. [this](#) (french)

- [Matplotlib](#) (scientific graphs)
- [NumPy](#) (array manipulations, linear algebra, Fourier transforms, random numbers,...)
- [SciPy](#) numerical methods (integrations, ODE, PDE,...)
- [SymPy](#) symbolis maths
- [Pandas](#), data analysis
- Pylab, combine Matplotlib, NumPy and SciPy
- [Scikit-learn](#), machine learning

## Fundamental numerical methods

- [Systems of linear equations](#)
  - Diagonalisation and triangularisation
  - LU decomposition : factorization in triangular matrices
- [Root findings : equations  \$f\(x\) = 0\$](#) 
  - Polynomial equations
  - Dichotomy
  - Secant method, Regula falsi
  - Newton-Raphson method
- [Numerical intégration](#) (integrals)
  - Simpson method and gaussian quadratures



### Learning outcomes :

- Systems of linear equations
  - failing of the theoretical way to solve a linear system using determinant and cofactors (np complexity)
  - triangularisation and diagonalisation principles : algorithm and complexity
  - “divide by zero” errors and pivot solutions
  - extension towards the matrix inversion
  - lower-upper LU decomposition and complexity ( $N^3$  for the decomposition step and  $N^2$  for substitution step). How to solve systems with varying independant vectors
  - special matrix require special algorithms : tridiagonal matrix algorithm (Thomas algorithm)
- Root findings
  - Bisection method (dichotomy) : simple and robust algorithm, invariant loop, slow convergence
  - iterative transformation  $x = f(x)$ , convergence and divergence situations
  - secant and regula falsi methods, Convergence Criterion of the Fixed Point Method
  - Newton-Raphson method (use of derivatives), quadratic convergence, failure, tolerance and stop condition
  - Van Wijngaardeb-Dekker-Brent method (“black box” in numerical packages)
  - Roots of polynomials and Bairstow's method
- Numerical intégration
  - Equally Spaced methods (trapezoidal, Simpson), accuracy, error dependance,...
  - Gaussian Quadratures and orthogonal polynomials (special integrals, scale transformations, error estimates,...)

## Classical numerical methods

### Ordinary\_differential\_equations (ODE)

#### Numerical solutions of ODE

- principe de discrétisation, méthode d'Euler
- Améliorations et méthodes de Runge-Kutta
- Runge-Kutta d'ordre 4
- Contrôle du pas d'intégration
- Méthodes predictor-corrector
- Méthodes d'extrapolation (Richardson, Burlish-Stoer)
- applications :
  - équations de cinétique chimique
  - Équation logistique
    -  [Modèle de Verhulst](#),  [dynamique des populations](#) et non-linéarité
    - Bifurcation, doublements de périodes et transition vers le chaos
  - Réactions chimiques oscillantes : Belousov-Zhabotinsky, Brusselator, Oregonator
  - Modèle proie-prédateur
  - Attracteur étrange, modèle atmosphérique de Lorenz

### Partial\_differential\_equations (PDE)

#### Numerical solutions of PDE

- Domaine d'application des équations : équation de diffusion, équation d'ondes, équations de Navier-Stokes
- Types de traitements numériques
- Différences finies et problèmes de diffusion
- Schémas classiques de différences finies
  - Résolutions stationnaires
  - Résolutions dépendantes du temps
- Méthodes explicites, critère de (ou d'in)stabilités et méthodes implicites

### Eigenvalues and eigenvectors

#### Eigenvalues and eigenvectors

applications à des problèmes de relaxation et de population, analyse de modes normaux de vibration, PCA (principal component analysis),...

### Non-linear systems of equations

- Newton-Raphson method

## Linear and non-linear least squares approximations

- Application to deconvolution (Levenberg-Marquardt algorithm)
- Références :
  - [An Open-Source, Cross-Platform Resource for Nonlinear Least-Squares Curve Fitting](#)  
Andreas Möglich, J. Chem. Educ., 2018, 95 (12), pp 2273–2278 DOI:  
10.1021/acs.jchemed.8b00649

## Chebyshev approximation

+ discussion of some approximations like  [Bhaskara I's sine approximation formula](#)

- <https://twitter.com/fermatlibrary/status/1267450081151782913>

## Molecules modelisation and visualization

### Minimization

Conformational problems

---

## Additional subjects

- Bioinformatics and related algorithm (biochemistry, mass spectrometry,...)
- Chemistry
  - quantum calculations, optimization, molecular mechanics
  - visualization, virtual reality
  - chemical informations on structures and reactions
- Data science, statistics (Python modules : Scipy, Pandas,...)
  - Time series analysis
  - Machine learning (Scikit-learn,...)
- Data visualization
  - boxplot, 3D, animations, graphs,...
- Sensors and interfaces, Arduino, Raspberry Pi, IoT
- Simulations
  - Agent base modelling and complex systems
  - cellular automaton
  - Simpy, active matter simulations...
- Digital image processing, image recognition
  - particle tracking,...

## References

- Gradient descent optimization

- [How to Implement Gradient Descent Optimization from Scratch](#) By Jason Brownlee on April 16, 2021
- Bioinformatics
  - [Biopython](#)
  - [Rosalind.info](#), learning bioinformatics by problem solutions
- Machine Learning
  - Scikit-learn
    - [Linear Regression in 6 lines of Python](#) (using scikit-learn)
  - [12 Algorithms Every Data Scientist Should Know](#)
- Deep Learning
  - TensorFlow
  - [Keras Tutorial: Deep Learning in Python](#)
- Arduino
  - [Fabrication of an Economical Arduino-Based Uniaxial Tensile Tester](#), Julien H. Arrizabalaga, Aaron D. Simmons, and Matthias U. Nollert, J. Chem. Educ., 2017, 94 (4), pp 530–533 DOI: 10.1021/acs.jchemed.6b00639
- PCA (principal component analysis)
  - [Learning Principal Component Analysis by Using Data from Air Quality Networks](#), Luis Vicente Pérez-Arribas, María Eugenia León-González, and Noelia Rosales-Conrado, J. Chem. Educ., 2017, 94 (4), pp 458–464 DOI: 10.1021/acs.jchemed.6b00550
  - [Principle Component Analysis in Python](#)
- K-Means
  - [K-means Clustering in Python](#)
- Applications (suggestions, examples,...)
  - [2D Ising Model in Python](#)
- chemistry
  - misc docs :
    - [Extraction of chemical structures and reactions from the literature](#)
  - [Animating Schrodinger Wave Function\( \$\psi\$ \) of a Particle Using Python \(with full code\) - Solving Particle in a Box Using Crank-Nicolson Method](#), Kowshik chilamkurthy, Medium, 02/03/2021
- Blog articles
  - [Simple Linear Regression Explained With Python - Explained in details which are easy to understand](#) Arjan de Haan, Medium, Jan 23 (+ [jupyter notebook](#))
  - [simpy : Simulate Real-life Events in Python Using SimPy](#) by Khuyen Tran, May, 2021, Medium, Towards Data Science
  - Active matter simulations (Vicsek, 1995) :
    - [Create Your Own Active Matter Simulation \(With Python\)](#) Philip Mocz, The Startup, Medium 2021 + <https://github.com/pmocz/activematter-python>
  - Fourier transforms : [Fourier Transforms: An Intuitive Visualisation - An intuitive visualization of discrete Fourier transforms applied to simple time-series data](#) Diego Unzueta, Towards Data Science, Medium, 17/09/2021

## Books

- [Solving Differential Equations in R](#), chez Springer, et en version électronique sur [SpringerLink](#)
- [Numerical Methods in Engineering with Python 3](#) 3rd Edition, Jaan Kiusalaas, 2013, Cambridge University Press, isbn: 9781107033856
- ...

## Jupyter notebooks

- [Jupyter notebooks and other materials developed for the Columbia course APMA 4300](#)
- [A Concrete Introduction to Probability \(using Python\)](#)

## MOOCs

- [Practical Numerical Methods with Python](#)

## Miscellaneous

- [Practical Numerical Methods with Python](#)
- <https://github.com/cfgnunes/numerical-methods-python> (?)
- [Presenting Code Using Jupyter Notebook Slides](#)
- <https://choosealicense.com>
- <https://creativecommons.org/choose/?lang=en>
- Published applications
  - [Introduction to Stochastic Simulations for Chemical and Physical Processes: Principles and Applications](#) Charles J. Weiss, Journal of Chemical Education 2017 94 (12), 1904-1910 DOI: 10.1021/acs.jchemed.7b00395
- [An overview of the Gradient Descent algorithm](#)
- [Building A Logistic Regression in Python, Step by Step](#)
- [Building Linear Regression in Python](#)
- [17 types of similarity and dissimilarity measures used in data science](#) : explains various methods for computing distances and showing their instances in our daily lives. Additionally, it will introduce you to the pydist2 package. Mahmoud Harmouch, Medium, 14/03/2021
- [Pythonic Perambulations](#) de Jake VanderPlas

---

## Synopsis (français)

(Méthodes de calcul appliqué à la chimie)

- Learning outcomes teaching unit (UE):
  - Apply standard numerical methods or existing software to solve numerical problems related to scientific research activities
  - Be active in the search for existing numerical methods adapted to problems encountered by chemists
- Content of the UE:
  - widespread methods : linear systems, numerical integration, root findings
  - Ordinary differential equations (numerical solutions, kinetic applications,...)
  - Partial differential equations (finite differences, diffusion problems)
  - Nonlinear systems of equations (Newton-Raphson method)
  - Eigenvalues eigenvectors problems (applications to relaxation and population problems)
  - Approximation by linear and non-linear least squares methods (maximum likelihood, application to deconvolution)
  - Chebyshev approximation

- Molecular modeling and visualization
- Minimization and conformational problems
- Prerequisite skills
  - Basic knowledge of a programming language
  - Basics of Mathematics
- Exercises and applications: codes written in or mainly written in Python, with the general libraries matplotlib, numpy, scipy, pandas as well as other specialized libraries, especially in chemistry
- Types of evaluations: Oral examination based on an in-depth study on one of the chapters of the course or an additional theme
- Acquis d'apprentissage UE :
  - Appliquer des méthodes numériques standards ou des logiciels existant pour résoudre des problèmes fondamentaux ou annexes, liés à des activités de recherche scientifique
  - Être actif dans la recherche de méthodes de résolution numérique existantes et adaptées à des problèmes auxquels les chimistes sont confrontés
- Contenu de l'UE :
  - Équations différentielles ordinaires (résolutions numériques et applications cinétiques)
  - Équations aux dérivées partielles (différences finies, problèmes de diffusion)
  - Systèmes d'équations non linéaires (méthode de Newton-Raphson)
  - Problèmes aux valeurs propres (applications à des problèmes de relaxation et de population)
  - Approximation par moindres carrés linéaires et non-linéaires (application à la déconvolution)
  - Approximation de Tchébyshev
  - Modélisation et visualisation de molécules
  - Minimisation et problèmes conformationnels
- Compétences préalables
  - Connaissance de base d'un langage de programmation
  - Bases des mathématiques
- Exercices et applications : codes écrits ou à écrire principalement en Python, avec les librairies générales matplotlib, numpy et scipy, ainsi que d'autres librairies spécialisées, notamment en chimie
- Types d'évaluations : Examen oral sur base d'un travail approfondi sur un des chapitres du cours ou un thème additionnel

## Pré-requis mathématiques

## Base de la programmation

## Méthodes numériques de base



- [Systèmes d'équations linéaires](#)
  - Diagonalisation et triangularisation
  - Décomposition LU en matrices triangulaires
- [Intégration numérique](#)
  - Simpson et quadratures gaussiennes
- [Résolutions d'équations du type  \$f\(x\) = 0\$](#) 
  - Équations polynomiales

- Recherche dichotomique
- Méthode de la sécante
- Méthode de Newton-raphson

## Méthodes numériques usuelles

### Équations différentielles ordinaires

#### Résolutions numériques des ODE

- principe de discrétisation, méthode d'Euler
- Améliorations et méthodes de Runge-Kutta
- Runge-Kutta d'ordre 4
- Contrôle du pas d'intégration
- Méthodes predictor-corrector
- Méthodes d'extrapolation (Richardson, Burlish-Stoer)
- applications :
  - équations de cinétique chimique
  - Équation logistique
    -  [Modèle de Verhulst](#),  [dynamique des populations](#) et non-linéarité
    - Bifurcation, doublements de périodes et transition vers le chaos
  - Réactions chimiques oscillantes : Belousov-Zhabotinsky, Brusselator, Oregonator
  - Modèle proie-prédateur
  - Attracteur étrange, modèle atmosphérique de Lorenz

### Équations aux dérivées partielles

#### Résolutions numériques des équations aux dérivées partielles

- Domaine d'application des équations : équation de diffusion, équation d'ondes, équations de Navier-Stokes
- Types de traitements numériques
- Différences finies et problèmes de diffusion
- Schémas classiques de différences finies
  - Résolutions stationnaires
  - Résolutions dépendantes du temps
- Méthodes explicites, critère de (ou d'in)stabilités et méthodes implicites

### Problèmes aux valeurs propres

#### Valeurs propres et vecteurs propres

applications à des problèmes de relaxation et de population, analyse de modes normaux de vibration, PCA (principal component analysis),...



## Systèmes d'équations non linéaires

Méthode de Newton-Raphson

## Approximation par moindres carrés linéaires et non-linéaires

application à la déconvolution

## Approximations de Tchébychev

## Modélisation et visualisation de molécules

## Minimisation

problèmes conformationnels

---

## Thèmes additionnels

- Bioinformatique et algorithmes spécifiques
- Chimie
  - calculs quantiques, de minimisation, de mécanique moléculaire
  - représentations
- Data science, statistiques (bibliothèque Python Pandas,...)
  - Time series analysis
  - Machine learning (Scikit-learn,...)
- Data visualization
  - boxplot, 3D, animations, graphes,...
- Capteurs et interface, Arduino, Raspberry Pi, IoT
- Simulations
  - Agent based modelling et systèmes complexes
  - Automates cellulaires
  - Simpy,...
- Traitement d'image
  - particle tracking,...

From:

<https://dvillers.umons.ac.be/wiki/> - Didier Villers, UMONS - wiki

Permanent link:

<https://dvillers.umons.ac.be/wiki/teaching:methcalchim:start?rev=1635151040>

Last update: **2021/10/25 10:37**

